Applying dynamic Bayesian networks in transliteration detection and generation

Nabende, Peter

Applying Dynamic Bayesian Networks in Transliteration Detection and Generation

Peter Nabende

rijksuniversiteit
groningen

Research presented in this thesis was carried out under the auspices of the Center for Language and Cognition Groningen in the Faculty of Arts at the University of Groningen. Research funding was from a second NUFFIC Uganda project (NPT-UGA-238) for strengthening ICT training and research capacity in public universities in Uganda.

Groningen Dissertations in Linguistics 95
ISSN 0928-0030

ISBN 978-90-367-5227-5

Document prepared with LaTeX 2ε and typeset by pdfTeX
Cover designer: Peter Nabende
Printed by Wörhmann Print Service, Zutphen, The Netherlands.

Rijksuniversiteit Groningen

Applying Dynamic Bayesian Networks in Transliteration Detection and Generation

Proefschrift

ter verkrijging van het doctoraat in de
Wiskunde en Natuurwetenschappen
aan de Rijksuniversiteit Groningen
op gezag van de
Rector Magnificus, dr. E. Sterken,
in het openbaar te verdedigen op
vrijdag 2 december 2011
om 16:15 uur

door

Peter Nabende
geboren op 18 maart 1980
te Mbale, Oeganda

Promotor:	Prof. dr. ir. J. A. Nerbonne
Copromotor:	Dr. ir. E. F. Tjong Kim Sang
Beoordelingscommissie:	Prof. dr. V. Baryamureeba Prof. dr. M. Biehl Prof. dr. K. U. Schulz

To my parents
and family

Acknowledgments

This thesis has been made possible with the additional help of people who contributed their ideas, time, and resources. It is very difficult for me to acknowledge all who were involved or those who feel were involved. Please accept my sincere apology if you feel you helped but have not been mentioned. I am grateful to all of you who helped even in the slightest form.

First of all, I am greatly indebted to my supervisors Jörg Tiedemann and Erik Tjong Kim Sang, and promoter John Nerbonne, whose ideas and comments helped shape many parts of this thesis. I am grateful to John Nerbonne as the main leader in the Netherlands of the NUFFIC II Uganda project that has funded all of my PhD research. For my research, John Nerbonne initiated the idea to use Dynamic Bayesian Networks (DBNs) for detecting pronunciation differences or for a related Natural Language Processing (NLP) task. And Jörg helped to suggest that alternative NLP task in machine transliteration which I would from then be involved in. Jörg also helped me in using weighted finite state automata and adapting some statistical machine translation approaches to machine transliteration which I used as baselines to compare with DBN models. Erik helped with software for a standard baseline approach of Pair n-gram models for detecting transliterations, and volunteered to translate the summary of this thesis to Dutch. Thank you again to Erik and John for the valuable comments and for making sure that my grammatical and spelling errors in the thesis were minimized as much as possible. I am grateful to Jörg for the first 18 months and to Erik for the other 30 months during which weekly meetings and updates about my progress were always eye-opening and made my PhD research experience exciting, and they helped to keep me focused and in the right direction.

I am also very grateful to the reading committee consisting of V. Baryamureeba, M. Biehl, and K. Schulz for accepting to read this thesis according to schedule and for their positive response.

Most of my study was centered around adapting Pair HMM and Dynamic Bayesian Network (DBN) software for computing transliteration similarity. I would like to thank Karim Filali for providing his Graphical Modeling Toolkit (GMTK) scripts for implementing DBN models and Martijn Wieling for the version of Pair HMM software that was used in dialect work. I extend my thanks to Svetlana Zinger for the helpful advise when starting to learn to use Pair HMM software. In dealing with Pair HMM software errors, I thank Martijn and Daniel de Kok for their help, and to Daniel again for suggesting the C/C++ class at University of Groningen's Donald Smits Center for Information Technology. Thanks to Martijn again for the informative overview on thesis printing.

From the Department of Information Science in the Faculty of Arts at University of Groningen, I would like to thank Gertjan van Noord for suggesting to use cross entropy for evaluating machine transliteration models. Thank you to Gosse Bouma for forwarding the first announcement about the Named Entities Workshop (NEWS) shared task on transliteration generation. The first shared task and two additional shared tasks would later become continuous points of motivation for my study. I am indebted to the organizers of the 2009 and 2010 NEWS for availing me with the standard transliteration datasets for various language pairs that I used in my experiments on applying DBN models in transliteration detection and generation. I am particularly grateful to Haizhou Li, A Kumaran, Min Zhang, Mitesh Khapra, and Vladmir Pervouchine.

During my final year, I enjoyed the company of Barbara Plank, Çagri Cöltekin, and Dörte Hessler who were also in the phase of compiling their PhD theses. I am grateful to the three for helpful suggestions from discussions on our thesis writing progress. I am grateful for their valuable help on Latex and the suggestion to use the tikz package in Latex which turned out to be perfect for most of the diagrams in my thesis. I also thank them for being remindful of various requirements including those that needed formalization and thesis printing details.

The department of Information Science in Groningen provided a wonderful environment to work in. I was very lucky to share an office with: Jörg, Raphael Aregu, Fridah Katushemererwe, and Gideon Kotzé from whom I got an idea about tree alignment and with whom I enjoyed and still enjoy online Chess games. Also, thank you to Gideon for accepting to be my paranimf and main contact for handling the printed copies of this thesis. In addition to my office mates and those I have already mentioned, this not so complete list includes some of the rest: Geoffrey Andogah, Proscovia Olango, John Kizito, Jackie Benavides, Harm Brower, Kostadin

Cholakov, Nynke van der Vliet, Peter Meindertsma, Tamás Biro, Leonie Bosveld-de Smet, George Welling, Valerio Basile, Johan Bos, Tim van de Cruys, Jelena Prókic, Ismail Fahmi, Jori Mur, Lonneke van der Plas. I am grateful for their contributions and the opportunity to learn from them during weekly reading group meetings. I also enjoyed playing soccer with most of them and many others that I can not mention all them here. Most of the people in this Department have been and continue to be an inspiration to me. I would like to thank Wyke van der Meer, the CLCG PhD coordinator during the four years of my study for the administrivia associated with my presence at the Faculty of Arts. I also extend my gratitude to all staff members of the CNL secretariat for their service. In the context of administering the NUFFIC II project in the Netherlands, I am very grateful to Erik Haarbrink and Gonny Lakerveld for taking care of my welfare in various aspects. Thank you too to Geertje Holtrop and Mark Strookappe.

I was lucky to meet many friends who made my stay in Groningen a wonderful experience. Most of them especially at the Bisschop Nierman Centrum International student house (Plutolaan) have enriched my life with pleasant memories. Many provided valuable information and help at the time I most needed it. Outside of Groningen, I am grateful to Eva and Mike Kagunda for the times we had together and especially for your incredible help.

Part of my study time was spent at Makerere University in Uganda. In Makerere University, I am very fortunate to be part of the School of Computing and Informatics Technology (SCIT). It is more likely that I would never have got the opportunity to pursue research presented in this thesis if I had not been part of SCIT. I am sincerely grateful to Venansius Baryamureeba for his main role in establishing SCIT and as the main leader in Uganda of the NUFFIC II Uganda project. I am also grateful to Jude Lubega as Makerere University coordinator of the same project and to Peace Tumuheki as one of the main administrators of the project. I would also like to thank Josephine Nabukenya as Dean of SCIT at the time of writing. Thank you too to Rossette Birungi for the responsibility to handle formal documents pertaining to my absence from SCIT. For that, I also thank Fausia Konde. I am still and will always be thankful to my MSc. Computer Science project supervisor Tom Wanyama who while at SCIT suggested to me the graphical models approach of Bayesian networks on which I would later base my interests for PhD studies. For my PhD research, I thank John Quinn for the discussions I always had with him about my progress and I am grateful for his invitations to present my research in the Machine learning research group at SCIT. I benefited from comments and questions in those presentations. Many colleagues at SCIT made my life comfortable including those I was lucky to share an office with (Godfrey Onyait, Noreda Kiremire, Paul Bakaki, Rose Nakibuule, Gilbert Maiga, and in the PhD room) and all those I interacted with during the lunch

breaks. I extend my gratitude to all those who helped.

Finally, I would like to thank my family for their everyday presence in my life and for the support they have provided throughout my entire life. I am deeply grateful to my parents Irene and Fred Wanziguya for making sure that I made it through primary and secondary school and for supplementing my undergraduate studies. Most of all, I am hugely indebted to them for funding my MSc studies without which the next step of pursuing PhD studies would still be a dream. What they sacrificed for me is not easy to repay. My siblings Richard Muloni, Carol Kwaga, James Zeere, Prima Zemei, Naume Muzaki, and Florence Wanyenze have always been a source of motivation and inspiration. They have been and will always be a blessing to me. I am also grateful to relatives close to this family who had a positive influence on me and who have helped me in my endeavours including while I was away in Groningen. Lastly but not least, I am specially grateful to my girlfriend Joyce for her love, patience, comfort and kind support.

May God bless you all.

Peter Nabende
October 22, 2011

Contents

List of Figures

List of Tables

Abbreviations

2-TBN	Two-slice Temporal Bayes net
ANN	Artificial Neural Network
ASR	Automatic Speech Recognition
CCE	Corpus cross entropy
CLIR	Cross Language Information Retrieval
CON DBN	Context-dependent Dynamic Bayesian network
CPD	Conditional probability distribution
CPT	Conditional probability table
CRF	Conditional Random Field
CVA	Cross validation accuracy
CVMRR	Cross validation mean reciprocal rank
DBN	Dynamic Bayesian network
EM	Expectation Maximization

FSA	Finite state acceptor
FST	Finite state transducer
GEM	Generalized expectation maximization
GMTK	Graphical modeling toolkit
HMM	Hidden Markov model
LCSR	Longest common subsequence ratio
MAP	Mean average precision
MCI DBN	Memoryless and context-independent Dynamic Bayesian network
MEM DBN	Memory-dependent Dynamic Bayesian network
MRR	Mean reciprocal rank
MT	Machine Translation
NE	Named entity
NEWS	Named entities workshop
NLP	Natural Language Processing
OCR	Optical character recognition
Pair HMM	Pair Hidden Markov model
PGM	Probabilistic graphical model
PSMT	Phrase-based Statistical Machine Translation
SMT	Statistical Machine Translation
TD	Transliteration Detection
WFSA	Weighted finite state acceptor
WFST	Weighted finite state transducer

Chapter 1

Introduction

1.1 Background

With the advent of the Web, various Natural Language Processing (NLP) systems including Machine Translation (MT) and Cross Language Information Retrieval (CLIR) are increasingly being accessed and used for cross language information processing. The automated NLP systems are useful as they help to overcome various limitations that are initially associated with manual information processing. Currently, a major benefit of using NLP systems is the instant generation of output given input data, and hence the possibility of processing and handling large amounts of data even with low cost computational resources. However, gains in processing capability of NLP systems are offset by poor output quality. There are many factors that can affect system output quality including the use of inadequate and error prone models in a particular application. This thesis is concerned with the case where NLP systems encounter words or phrases in data that are *unknown* to them which complicate system processing and consquently result in poor output quality.

In bi-lingual or multi-lingual applications, the problem caused by *unknown* words is mainly complicated by lack of a corresponding representation in some *target language*. Table 1.1 illustrates this problem where a state-of-the-art Web-based machine translation system (Google Translate[1]) fails to get corresponding representations in the Chinese language for the two underlined words (*Falade* and *Fansidar*) written using the Latin alphabet and in an English sentence. As Table 1.1 shows, the strategy used by the MT system and which is commonly used by similar systems in dealing with *unknown* words is to simply copy them to the resulting output. Table 1.1 also shows that all the words that the system failed to represent in the Chinese translation are names (*Falade* is a person name whereas *Fansidar* is a drug name). This strategy

[1] http://translate.google.com

English input	Chinese translation
While the battle against malaria is gradually being won according to Dr. Falade, the use of Fansidar as a combination drug is highly discouraged.	儘管防治瘧疾的戰鬥正在逐贏得根據博士Falade，使用的Fansidar作為一個組合藥物是非常氣餒。

Table 1.1: *A Web-based MT system's English-to-Chinese translation*

of simply copying *unknown* words may be useful in cases where the source and target language use the same writing system since there is a likelihood to retain spellings for named entities across the languages. When the source and target language use different writing systems, the strategy of copying *unknown* words is not useful.

Transliteration, a process used to convert *new* words from a *source language* to a phonetically equivalent, understandable, and representable form using the writing system of the *target language* is currently the most natural approach to dealing with *unknown* words for the case where different writing systems are used. For example, a suggestion of a phonetically equivalent representation for the word *Fansidar* in Table 1.1 could be like this in Chinese: 反思大 /fan-si-da/. Here, the main NLP system would require an additional transliteration sub-system that helps generate hypothetical target language representations for any identified *unknown* word. A different approach to employing transliteration in a cross-language processing system is to complement the system's bi-lingual lexicon with a separately acquired *transliteration-pair lexicon*. Either way, transliteration is currently important both as a topic and as a sub-task in NLP since system effectiveness is expected to increase when it is used.

In both of the transliteration-based approaches in the last paragraph, various methods have been proposed and used to help improve the quality of the *system generated transliterations* or *transliteration-pair lexicons*. However, recent work, for example the shared tasks on transliteration mining (Kumaran et al. 2010b) and transliteration generation (Li et al. 2010), shows that there is need to identify more approaches that can help improve system performance in the two tasks. Research on using a given method in each of the two transliteration-related tasks is usually based on an interest of attaining improvements in system performance.

In this thesis, the interest is to see whether models derived from two edit distance-based Dynamic Bayesian Network (DBN)-modeling approaches can lead to improvements in transliteration detection and generation. The two edit distance-based approaches implement Pair Hidden Markov Models (Pair HMMs) and transduction-based DBN models. The Pair HMM approach originates from work in biological sequence analysis (Durbin et al. 1998) and was later adapted to compute word similarity and successfully applied in cognate identification(Mackay and Kondrak 2005,

Kondrak and Sherif 2006) and in dialect comparison (Wieling et al. 2007). The transduction-based DBN approach originates from work in automatic speech recognition with successful applications in the tasks of pronunciation classification (Filali and Bilmes 2005) and cognate identification (Kondrak and Sherif 2006). The motivation to investigate the two approaches is based on our knowledge about their successful application in tasks with requirements similar to those for transliteration detection. While automated transliteration detection and generation still demand new approaches, there was not yet any investigation about the use of the two edit distance-based DBN modeling approaches in transliteration detection and generation.

We investigate the application of several edit distance-based DBN models associated with the two approaches. Preliminary work involves an empirical investigation on the application of different Pair HMMs in detecting transliteration pairs using 'simple' datasets where the source and target language texts are based on the Latin alphabet. We use text based on the Latin alphabet not only because it is simple to process computationally, but because we wanted to first determine the feasibility of having Pair HMMs model different character representations of written words using the easy to process data. We have assumed bi-lingual text based on the same alphabet to be used as transliteration data because of different written representations for similar pronunciations between source and target languages and the use of different diacritical characters to convey additional pronunciations for some languages. In this case, our 'simple' datasets comprise of geographic name pairs extracted from the Geonames[2] database for three language pairs: English-French, English-German, and English-Dutch. We also extracted English-Russian geographic name pairs to form our first experimental dataset where the languages use different writing systems. This English-Russian dataset is part of the preliminary setup. We then use more datasets obtained from the 2009 (Li et al. 2009) and 2010 (Li et al. 2010) shared tasks on transliteration generation for a further evaluation of the Pair HMMs and transduction-based DBN models in another experimental Transliteration Detection (TD) setup. The data from the shared tasks for this set of experiments includes the following seven language pairs: English-Bengali, English-Chinese, English-Hindi, English-Kannada, English-Russian, English-Tamil, and English-Thai. For this set of experiments, we also evaluate the DBN models against a standard baseline approach of using cross-language Pair n-gram information for transliteration detection and determine best performing DBN models for later evaluation on real-world data. In order to establish the value of applying the proposed DBN-related approaches in transliteration detection and generation on real-world data, we evaluate the application of some of the Pair HMMs and transduction-based DBN models against state-of-the-art methods that were used in the 2009 and 2010 shared tasks on transliteration gener-

[2]http://www.geonames.org

ation (Li et al. 2009, Li et al. 2010) and in the 2010 shared task on transliteration mining (Kumaran et al. 2010b). Here, the comparison against state-of-the-art methods is simplified by using the same datasets and same evaluation setup as used during the shared tasks. The shared task data used in the transliteration mining experiments comprises of standard Wikipedia topic pairs for five language pairs: English-Arabic, English-Chinese, English-Hindi, English-Russian, and English-Tamil.

1.2 Research goal

The overall research goal in this thesis is to apply Pair Hidden Markov models (Pair HMMs) and transduction-based Dynamic Bayesian Network models in transliteration detection and generation while aiming for improvements over existing techniques. Based on this goal, the thesis aims to address the following questions:

1. Does the current state of research necessitate an investigation into the use of new methods (such as the Dynamic Bayesian Network (DBN) models proposed in this thesis) for transliteration detection and generation?
   For this question, we would like to know whether existing methods for transliteration detection and generation suffice.

2. Can DBN models that have been used in tasks (such as cognate identification and pronunciation classification) with requirements similar to transliteration detection be valuable when used in the context of computing transliteration similarity? Related to that, can modifications to these DBN models that meet the requirements for computing transliteration similarity be valuable in the detection of transliteration pairs?
   Here, we would like to know whether the assumptions associated with the successful application of DBN models in previous tasks could also lead to successful application of the DBN models in transliteration detection and generation.

3. What features are critical to the use of DBNs for modeling transliteration similarity?
   In the thesis, we investigate various DBN model structures and parameter settings. We would like to know which types of Pair HMMs and transduction-based DBN models adequately address factors that are important in modeling transliteration similarity.

4. Can the application of DBN models improve transliteration detection and generation quality as compared to current state-of-the-art methods?
   Results obtained from representative experimental setups may not portray the true effect of applying DBN models on real-world data. Here, we are interested

in knowing whether there can be any benefits of using DBN models when evaluated on real-world data for transliteration detection and generation. This, as the question has been put, requires an evaluation against state-of-the-art methods that would be applied in a similar manner.

1.3 Research approach

The research approach used in this thesis is mostly focused on addressing the questions in the previous section. To address the first and second questions, I undertook an exhaustive literature review on approaches that have been proposed and used in previous work for transliteration detection and generation. In order to avoid repetitions, I also considered recent comprehensive literature reviews on machine transliteration, for example in Karimi et al (2011). The literature review is mostly exploratory, but a critical analysis is given where suitable.

To address the remaining questions, we follow an empirical approach in which experiments are conducted to evaluate the performance of several DBN models in transliteration detection and generation. For the third and fourth question, we define an experimental transliteration detection setup to evaluate the use of proposed DBN models for computing transliteration similarity. Here, we first experiment with our own prepared transliteration data from a Web-based geographic names database (Geonames) and later we experiment with standard transliteration corpora from the 2009 and 2010 shared tasks on transliteration generation (Li et al. 2009, Li et al. 2010). Each dataset has been manually verified and we assume that each source language word has exactly one target language word match. We use the experimental transliteration detection setup at this stage to identify DBN models that could be useful for transliteration generation and in detecting transliteration pairs from 'noisy' real-world data.

For the fourth research question, our participation in both the 2009 and 2010 NEWS shared tasks on transliteration generation and transliteration mining respectively ensured an evaluation of the application of the proposed DBN models against state-of-the-art methods that were also applied on the same standard transliteration corpora. In the thesis, I have followed the same evaluation setups as specified for the NEWS 2009 shared task on transliteration generation (Li et al. 2009) and for the NEWS 2010 shared task on transliteration mining (Kumaran et al. 2010b) for evaluating the DBN models. For the transliteration generation task, participating teams were supplied with training and development data for a dozen language pairs to be used for training and tuning the participating systems. After training and development, each participating team was required to submit ten system generated candidate transliterations per source language word in the the test set for a given lan-

guage pair. For the transliteration mining task, each participating team was availed with a seed set of matching name pairs for five language pairs to be used as initial training data. In these experiments we use models that performed well in the experimental transliteration detection setup mentioned in the previous paragraph.

1.4 Overview of the rest of the thesis

In Chapter 2, we present a literature review on transliteration detection and generation aimed at determining the current state of research on the two tasks and the need for new solutions to some identified gaps. We give a general view for each of the two tasks and a description of the main phases. We then review several modeling approaches that have been used in each task ranging from the earliest to the current state-of-the-art. This also involves an analysis of performances achieved by the approaches in the two tasks.

In Chapter 3, we introduce the main concepts underlying the framework of Dynamic Bayesian Networks (DBNs). First, we introduce Bayesian Networks from which DBNs are an extension. We discuss BNs and subsequently DBNs according to three aspects: their representation, inference, and learning methods. For DBN learning in particular, we review a theoretical explanation of the Expectation Maximization (EM) algorithm. The EM algorithm and its generalized form are applied in different ways to train all DBN models that we have proposed for transliteration detection and generation. At the end of the chapter, we specify the general framework for applying the DBN models in transliteration detection and generation.

In Chapter 4, we introduce the Pair HMM approach as the first of the DBN methods proposed for use in transliteration detection and generation. First, we provide some background regarding the origins of the Pair HMM method from its inception in the field of biological sequence analysis (Durbin et al. 1998) to its adaptation for estimating word similarity(Mackay and Kondrak 2005). A discussion then follows of the requirements that need to be met in order to adapt the Pair HMM approach for estimating transliteration similarity. Different plausible Pair HMM parameterization settings are proposed and evaluated in an experimental transliteration detection (TD) setup. We first investigate two settings for the Pair HMMs in the experimental TD task: in the first setting, we assume that only one character vocabulary is used to generate the source and target words, and in the second setting, we assume that the Pair HMMs use separate character vocabularies corresponding to the source and target language writing systems. It is obvious that the second setting relates more with transliteration, and its related experiments are aimed at determining the necessity to model the differences in the source and target language vocabularies in Pair HMM emission parameters which are used for computing transliteration similarity.

Here, we use four data sets obtained from the Geonames database as described in the Background section (1.1). We then investigate the use of different Pair HMM string similarity scoring algorithms and the use of different definitions for Pair HMM transition parameters. For this investigation, we used standard corpora from the NEWS 2009 and NEWS 2010 shared tasks on transliteration generation. The transliteration detection performance of the Pair HMM approach is evaluated against that of a standard baseline approach of using 'pair n-gram' models.

In Chapter 5, we introduce the transduction-based Dynamic Bayesian Network approach as the second DBN-related approach we have proposed to apply for computing transliteration similarity. First, we review the approach as initially proposed by Filali and Bilmes (2005) in their pronunciation classification task. A discussion then follows regarding the adaptation of the transduction-based DBN modeling approach in the context of transliteration similarity estimation. In proposing different transduction-based DBN model generalizations, we start with a presentation of an approximate transduction-based DBN model representation for the Pair HMMs. We have successfully adapted three DBN model generalizations associated with the transduction-based DBN modeling approach for computing transliteration similarity. These are described in addition to the baseline DBN model template from which they are derived. Each of the DBN model generalizations is used to account for specific types of temporal dependencies, including: dependencies that capture memory from previous edit states of a DBN model; contextual dependencies of edit states of a DBN model on either source or target string elements or on elements from both source and target strings; and dependencies that account for the lengths of edit steps needed for string similarity estimation. We then investigate the use of several models from the DBN model generalizations in the experimental transliteration identification task introduced in Chapter 4 using standard transliteration corpora from the shared tasks as mentioned in Chapter 4. The performance from the use of the transduction-based DBN models is evaluated against that of the baseline pair n-gram approach and the best performing Pair HMMs in Chapter 4. Our analysis of the results leads us to further propose and test several ensembles of DBN models for computing transliteration similarity.

In Chapter 6, we present an evaluation of the use of the DBN models in mining transliterations from real-word data (specifically, from the Web-based Wikipedia encyclopedic resource). Two transliteration mining sub-tasks for evaluating the DBN models are first introduced. In the first sub-task, we use the same evaluation set-up as that used in the NEWS 2010 shared task on transliteration mining where participating systems are evaluated on mining transliterations from paired cross-language Wikipedia topics. For this task, we evaluate the Pair HMM and transduction-based DBN methods against state-of-the-art methods that were used by the other partici-

pating teams in the shared task. In the second sub-task we propose and evaluate the application of the DBN models on paired cross-language Wikipedia article content in addition to using the respective paired Wikipedia topics.

In Chapter 7, we present an evaluation of the use of Pair HMMs in transliteration generation. Although Pair HMMs were initially proposed just for the purpose of computing string similarity – where they have been successfully used – this chapter is aimed at determining whether the Pair HMMs could as well be valuable in transliteration generation. Two transliteration generation sub-tasks are first introduced. In the first sub-task, we use the same evaluation setup as that used in the NEWS 2009 (Li et al. 2009) and NEWS 2010 (Li et al. 2010) shared tasks on transliteration generation. In the second sub-task we propose using the transliteration generation framework for translating named entities between languages that use the same writing system. We then describe a scheme for representing Pair HMM parameters as parameters in weighted finite state automata to allow for their use in transliteration generation. We also describe various types of other weighted finite state automata for evaluation in addition to the Pair HMM-based models. For the first task, we report on results associated with the use of weighted finite state automate including the Pair HMM-based models and compare them to results associated with the use of a state-of-the-art phrase-based statistical machine translation approach. For the second task, we evaluate the weighted finite state automata models and the phrase-based statistical machine translation approach against the standard baseline of copying *unknown* words.

Chapter 8 concludes the thesis with a discussion of results on the application of the two proposed DBN-related approaches in the two tasks of transliteration mining and generation. We also point out the contributions of the thesis. Finally, we present our suggestions for future work including work that we have not managed to cover in the thesis.

Chapter 2

A review on machine transliteration

2.1 Introduction

The origins of the use of transliteration as a process for dealing with unknown words in a foreign language or in a dialect of a particular language seem to be non-existent in transliteration literature. However, the systematic attempts to create systems for representing characters in a writing system of origin (for example in Japanese or Chinese) to characters in a different language using its writing system (for example to English using the Latin alphabet) are well documented and these systems are commonly referred to as 'transliteration systems'. The term *Romanization* is often associated with transliteration systems where the Roman alphabet is used to represent characters from a different writing system. For example, the American Library Association-Library of Congress (ALA-LC) romanization tables[1] constitute one of the largest collection of romanization systems for representing text in more than 150 languages written in various non-Roman scripts using the Latin alphabet. Contemporary literature suggests that transliteration is likely to have started as a process for converting a given word in a language of origin to a phonetically equivalent, understandable, and orthographically representable form in some target language (Li et al. 2009). Specifically, if the conversion is from a language of origin, the process is called *forward transliteration*. *Backward transliteration* or *reverse transliteration* is defined as the reverse process, where the aim is to find the original word representation in a language of origin given an existing word in a foreign language (for example finding the original Russian name (дмитрий) given an English name variant 'Dmitriy'.

The use of automated Natural Language Processing (NLP) applications such as Machine Translation (MT) and the non-diminishing importance of *unknown* words

[1] http://www.loc.gov/catdir/cpso/roman.html

that these applications encounter necessitated the use of automated methods as well to help deal with them. The most popular reference to one of the earliest attempts at fully automating the process of transliteration in the sense of processing named entities (which constitute the largest percentage of *unknown* words) dates back to almost two decades ago (Arbabi et al. 1994) where a combination of a rule-based expert system and an artificial neural network were used for including vowels in Arabic names. Also, around just the same time, studies on the automated search for named entities within a language and across languages had already started. Currently the general term used for cross-language named entity search for the case where the languages use different writing systems is *transliteration mining*. From the later half of the 1990s on, various approaches have been proposed to handle named entities (NEs) in cross language applications with regard to both the converting of NEs from one writing system to another, and to the search of corresponding NEs in different writing systems. In this chapter, we introduce the current view of the two tasks: *transliteration generation*, and *transliteration detection*, and with respect to each task, we review some of the major approaches that have been used from the earliest to current state-of-the-art. The organization of our review will follow the same order of presentation for the two tasks throughout the thesis.

Notation

To simplify our review of the various transliteration detection and generation approaches, we establish some notation that is common to most of the methods. The transliteration process in both detection and generation involves an analysis of the source and target language words which we denote here as: S for a source language word and T for a target language word. However, we shall extend the notation whenever there is need to reflect the point of discussion. For example, it may be necessary to specify a source word constituting m characters as $S_1^m = s_1 s_2 ... s_m$ and a target word with n characters as $T_1^n = t_1 t_2 ... t_n$. When discussing a phonetic-based approach we use SP to denote the phonetic representation of a source word, while TP is used to denote the phonetic representation of the target word. For the constituent phonetic units, we use $SP_1^l = sp_1 sp_2 ... sp_l$ for the source word and $TP_1^k = tp_1 tp_2 ... tp_k$ for the target word. For other specific representations, additional notation will be defined as per the need.

2.2 Transliteration Detection

The process of detecting transliterations generally involves the search for corresponding NEs from a collection of candidate NEs between two or more languages in different

writing systems. The main differences in the transliteration detection approaches are associated with: the data source for obtaining candidate NEs in each of the languages; the methods that are used to identify candidate NEs; and the methods that are used to relate and extract transliteration pairs. We shall review transliteration detection approaches based on the type of data resource used. For each type of data resource, we present some examples from transliteration literature and the subsequent methods that are used for candidate NE identification, transliteration similarity estimation, and transliteration pair extraction.

2.2.1 Data resources and transliteration detection methods

Transliteration detection necessitates the use of a bi(multi)-lingual[2] corpus from which we expect to match bi-lingual NEs. That is, the resource when considered as a whole, should have a reasonable amount of text in at least two or more languages to enable the identification and extraction of similar words across different languages. The most common types of bi-lingual data resources for transliteration detection can be classified into: bi-lingual single document texts, which consist of texts in two or more languages in the same document; parallel corpora, which consists of texts in two or more languages and where the texts are translations of one another (Karimi et al. 2011) and in different documents where corresponding sentences that are related through a given identifier are exact translations of each other; and comparable corpora, which is text in two or more languages and in different documents where the corresponding text are not exact translations of each other (as is the case in parallel corpora). The main differences in the transliteration mining process for the different approaches are associated with the kind of data resource that is used and the identification of candidate NEs from a given data resource. After the identification of candidate NEs, the setup for comparing and extracting NEs across different languages is often similar. In the following we review a selection of some examples and transliteration detection methods associated with each of the different types of bi-lingual data resources.

a) Bi/Multi-lingual single document text

The use of single document text for detecting transliteration pairs usually requires the application of prior knowledge about the presentation of different entities in a given document. Based on how bi-lingual text is represented, the transliteration detection

[2]Although the term *multi-lingual* is a generalization of *bi-lingual*, most of the approaches utilize *bilingual resources*. We therefore prefer to use the term *bi-lingual* in a general discussion to represent both cases. However, when describing a given approach that uses *multi-lingual resources* (that is, in more than two languages), we will correctly specify it as a *multi-lingual resource*.

process may be simplified or may require some additional pre-processing steps before applying a transliteration similarity estimation method. In the examples below, we see two different representations. In one example, source words are hypothesized to exist in parentheses next to target words in a sentence (Lin et al. 2004). In the other example (Kuo et al. 2007), source words are hypothesized to collocate with target words in a sentence but there is rarely an existence of delimiters that enclose source words.

Kuo et al. (2007) use predominantly Chinese Web pages, where transliterated words are collocated closely with their original source(English) words and the source words are often appositives of neighboring target(Chinese) words in a *close context*. They assume that the scope of a *close context* is within a sentence boundary which is delimited by punctuation such as full stops, commas, question and exclamation marks; and that it is a range of proximity where a source word and its target transliteration collocate. Kuo et al. also suggest that in cases where there are different types of words in a *close context*, we need to consider only word pairs that are most likely to be associated with phonetic transliteration. To describe how candidate named entities are identified, we use one of their examples, which illustrates collocations of source and target language words:

"...經營Kuro 庫洛P2P 音樂交換軟體的飛行網, 3 日發表P2P 與版權爭議的解決方案 C2C (Content to Community)..."

In the example above, "庫洛 /KU-LUO/" is a transliteration for "Kuro", the two can also be qualified as a candidate transliteration pair. However, although C2C is collocated with "Content to Community", the latter is just an acronym expansion and not a transliteration; such a pair can not be used as a candidate transliteration pair. Based on this observation, Kuo et al. propose a procedure for identifying candidate pairs which is as follows: 1) the predominantly Chinese Web page is segmented into sentences using punctuation marks as delimiters; 2) a search is made for any source language word S in each sentence; 3) if an English word S is recognized, then a *k-neighborhood* is defined to serve as the *close context* of the recognized English word; 4) $T \in \Omega$ is defined as a target transliteration candidate in the *k-neighborhood*, where Ω is the set of all transliteration candidates in the *k-neighborhood*. In the example above, "Kuro" is recognized as an English word, "經營 /JIN-YIN/" and "庫洛 /KU-LUO/" are suggested in a *close context*, the left and right k-neighborhoods. Two candidate pairs, "Kuro-經營" and "Kuro-庫洛" are then selected for further examination. For transliteration similarity estimation, Kuo et al. (2007) use a phonetic similarity (PS) modeling approach. The candidate words are first transformed into syllabic sequences Ssy for the English word and Tsy for the Chinese word. The PS model is then used to identify the most probable *Tsy'* that matches *S*. In the PS model, they formulate

their transliteration process using the noisy channel model (Brown et al. 1993) and by applying Bayes' rule, $P(T|S)$ is expressed as:

$$P(\mathrm{T}|\mathrm{S}) = \frac{P(\mathrm{S}|\mathrm{T})P(\mathrm{T})}{P(\mathrm{S})} \tag{2.1}$$

where $P(\mathrm{S}|\mathrm{T})$ represents the noisy channel probability (also called transliteration probability) and $P(\mathrm{T})$ is the language model probability. $P(\mathrm{S}|\mathrm{T})$ is approximated using a phonetic confusion probability $P(\mathrm{Ssy}|\mathrm{Tsy})$ which is obtained from a phonetic confusion probability matrix. They propose three ways of estimating the syllable-based confusion matrix: 1) $P_{\mathrm{ASR}}(\mathrm{Ssy}_m|\mathrm{Tsy}_n)$ for which an automatic speech recognition (ASR) system is used and where a labeled English speech database is run through a Chinese ASR system; 2) $P_{\mathrm{SYL}}(\mathrm{Ssy}_m|\mathrm{Tsy}_n)$ for which a Syllable PSM is used and where the syllable confusion probability is estimated by extracting transliteration pairs which are converted to syllables and to phonemes; 3) $P_{\mathrm{SS}}(\mathrm{Ssy}_m|\mathrm{Tsy}_n)$ for which a Sub-syllable PSM is used and where the syllable confusion matrix is estimated using sub-syllable confusion probability. Kuo et al. exploit the three confusion matrices in different stages for transliteration similarity estimation. After obtaining a similarity score and ranking the candidate list of T, they identify the most probable Tsy' by using a hypothesis test to decide whether T' is a transliteration of S.[3]

Lin et al. (2004) use both single document bi-lingual text and parallel text for extracting English Chinese transliterations. We present their approach for the single document text here and that for the parallel text in the next subsection on *Parallel corpora*. For the single document bi-lingual text, they exploit the fact that some data resources print source language terms in parentheses following their transliterations as shown in their example below:

國西部城市(1995年人口約247,000),位於科隆(Cologne)西北方 ...

In the example above 科隆 is a transliteration of Cologne. During transliteration similarity estimation, Lin et al. use a statistical transliteration model. The source language word (S) is first split into k transliteration units (TUs) $\mathrm{S} = su_1, su_2, ..., su_k$ which are then converted independently into k target characters $tc_1, tc_2, ..., tc_k$ using the statistical transliteration model. The tc_j's are subsequently combined to produce a target word T. Their transliteration model for $P(tc_j|su_i)$ is estimated using an Expectation Maximization (EM) algorithm with Viterbi decoding.

[3]Kuo et al. (2007) provide a much more detailed description of their approach with many mathematical formulations, but because of space constraints, we have decided to omit most of the details.

b) Parallel corpora

The use of parallel corpora necessitates the alignment of sentences between two languages. As will be seen below, some approaches apply a sentence alignment procedure as part of the transliteration mining process whereas other approaches use data resources in which the sentences are already aligned. Each aligned sentence pair is then hypothesized to contain similar NEs across the different languages. Most of the approaches go a step further in filtering out unnecessary entities in at least one of the languages before applying a transliteration similarity estimation method.

Sherif and Kondrak (2007a) use two sets of sentence aligned bi-text from an *Arabic tree bank part 1-10k word English* translation corpus, and an *Arabic English parallel news Part 1* corpus. They report that the two corpora contain Arabic news articles and their English translations aligned at the sentence level. They use the tree bank data as development data to optimize the acceptance threshold used by one of their methods for transliteration similarity estimation and extraction. They use the following pre-processing procedure to identify candidate NEs. First, they tokenize the English corpus using a tokenization tool (Word splitter). After tokenization, they remove all uncapitalized words; stop words are also removed from both sides of the bi-text. Lastly, English words of length less than 4 and Arabic words of length less than 3 are removed. Sherif and Kondrak (2007a) then apply a number of models for word similarity estimation including the bootstrapped stochastic transducer which is their main proposed method in the paper. Below, we summarize the similarity estimation methods that Sherif and Kondrak use:

1) Levenshtein edit distance (LED). Is used as the baseline method. To enable the computation of LED, a common representation between the source and target languages is needed. Specifically, Arabic candidate NEs are romanized to get to the common representation.

2) ALINE, is a phonetic-based word similarity estimation algorithm where individual phonemes input to the algorithm are decomposed into a dozen phonetic features, such as Place, Manner, and Voice. Then, a substitution score between a pair of phonemes is based on the similarity as assessed by a comparison of the individual features. After an optimal alignment of the two words is computed with a dynamic programming algorithm, the overall similarity score is set to the sum of the scores of all links in the alignment normalized by the length of the longer of the two words. The source and target words are first converted into phonetic transcriptions using a deterministic rule-based transformation.

3) Fuzzy string-matching algorithm. This was initially proposed by Freeman et al. (2006). The Fuzzy matching algorithm is based on the Levenshtein Edit Distance but encodes more knowledge about the relationships between the source and target

language. In this case, the LED is computed using letter equivalences as matches instead of identities. The source and target language letters within a class are treated as identities. The resulting Levenshtein distance is then normalized by the sum of the lengths of both words.
4) The main model proposed in Sherif and Kondrak (2007a) is a stochastic transducer from Ristad and Yianilos (1997) which is trained iteratively, and then applied to score a pair of candidate NEs.

In a rather different NE identification approach, Lee and Chang (2003) first apply a sentence alignment procedure to align parallel texts at the sentence level. An NE tagger is used to identify proper nouns in the source text (English) which serve as candidate source NEs for identifying transliterations in the target language (Chinese). Lee and Chang (2003) also formulate the transliteration problem as a noisy channel model while exploiting phonetic similarities between source words (S) and target words (T). The computation for $P(\mathrm{T}|\mathrm{S})$ is first formulated as a marginalization over an alignment sequence (δ):

$$P(\mathrm{T}|\mathrm{S}) = \sum_{\delta} P(\mathrm{T}, \delta|\mathrm{S}) = \sum_{\delta} P(\mathrm{T}|\delta, \mathrm{S})P(\delta|\mathrm{S}). \tag{2.2}$$

where δ represents an alignment candidate with $\delta = \delta_1, \delta_2, ..., \delta_{\mathrm{N}}$ match types. To reduce computational complexity, the summation criterion in Equation 2.2 is changed into a maximization and $P(\mathrm{T}|\mathrm{S})$ is approximated as:

$$P(\mathrm{T}|\mathrm{S}) \approx \max_{\delta} P(\mathrm{T}|\delta, \mathrm{S})P(\delta|\mathrm{S}) \approx \max_{\delta} P(\mathrm{T}|\delta, \mathrm{S})P(\delta) \tag{2.3}$$

Using transliteration units su_1^{N} for the source word and tu_1^{N} for the target word, Lee and Chang re-approximate $P(\mathrm{T}|\delta, \mathrm{S})P(\delta)$ in Equation 2.3 as follows:

$$P(\mathrm{T}|\delta, \mathrm{S})P(\delta) = P(tu_1^{\mathrm{N}}|su_1^{\mathrm{N}})P(\delta_1, \delta_2, ..., \delta_{\mathrm{N}}) \approx \prod_{i=1}^{\mathrm{N}} P(tu_i|su_i)P(\delta_i). \tag{2.4}$$

Finally $\log P(\mathrm{T}|\mathrm{S})$ is computed as:

$$\log P(\mathrm{T}|\mathrm{S}) \approx \max_{\delta} \sum_{i=1}^{\mathrm{N}} (\log P(tu_i|su_i) + \log P(\delta_i))$$

The maximum accumulated *log probability* among all possible alignment paths is computed using a dynamic programming procedure. Lee and Chang (2003) estimate the model probabilities using an Expectation Maximization (EM) procedure. They also incorporate some linguistic processing in their method, first to accelerate the convergence of EM training and then during transliteration similarity estimation to improve transliteration identification quality.

Lin et al. (2004) use a named entity identification approach similar to Lee and Chang's (2003) approach above. We already saw in the previous section (on *Bi/Multilingual single document text*) that Lin et al. (2004) use both the single document bilingual text and parallel text to mine transliteration pairs. For parallel text, Lin et al. first identify proper names in the source (English) sentence and then subsequently identify transliterations for each proper name. They suggest the use of a part of speech tagger and named entity recognizer for identifying English proper nouns. All words in the target language (Chinese) sentence are considered as transliteration candidates. They then use Viterbi decoding to identify the transliterations in the target language sentence using the same procedure as described at the end of the previous subsection.

c) Comparable corpora

Comparable corpora is mainly obtained from Web-based cross-language text, usually with the aid of cross-language links. Common sources include: time-aligned news articles over a given period of time, and encyclopedic resources such as Wikipedia. Most of the recent work in transliteration mining use comparable corpora as they are easy to acquire, and they can be used without the need of applying a sentence alignment procedure as is the case for parallel corpora. In this case, we simply use similar text with the assumption that they are likely to contain a reasonable number of corresponding NEs between the source and target languages. In the following, we review some of the recent approaches that use comparable corpora.

Udupa et al. (2009) use comparable news corpora for mining NE transliteration equivalents. They investigate the effectiveness of using different sets of paired documents for mining transliteration equivalents. In one stage, they investigate the use of documents comprising the comparable corpora $(\mathcal{C}_s, \mathcal{C}_t)$, and in another stage, they investigate the use of paired documents $(A_{s,t})$ obtained from $(\mathcal{C}_s, \mathcal{C}_t)$ as highly similar documents. In the latter stage, they use a cross-language document similarity model (KL-divergence) to estimate content similarity between documents $(\mathrm{D}_s, \mathrm{D}_t)$. Using V_s to denote the source language vocabulary and V_t the target language vocabulary, Udupa et al. compute KL-divergence as follows:

$$-\mathrm{KL}(\mathrm{D}_s|\mathrm{D}_t) = \sum_{\mathrm{T}\in\mathrm{V}_t} P(\mathrm{T}|\mathrm{D}_s) \log \frac{P(\mathrm{T}|\mathrm{D}_t)}{P(\mathrm{T}|\mathrm{D}_s)} \tag{2.5}$$

and since the interest is in target documents which are similar to a given source document, the denominator in Equation 2.5 (which is independent of the target document) can be ignored. By expanding $P(\mathrm{T}|\mathrm{D}_s)$, the following Equation is used for computing cross-language similarity:

$$\sum_{\mathrm{T}\in\mathrm{V}_t}\sum_{\mathrm{S}\in\mathrm{V}_s} P(\mathrm{S}|\mathrm{D}_s)P(\mathrm{T}|\mathrm{S})\log P(\mathrm{T}|\mathrm{D}_t).$$

Udupa et al. then use two transliteration similarity models on each document pair $(\mathrm{D}_s, \mathrm{D}_t)$ in $A_{s,t}$ to produce a set $\mathrm{Pairs}_{s,t}$ of NE transliteration equivalents. The first model, which they call the discriminative transliteration model uses a logistic function to compute transliteration similarity between every candidate pair of words (S,T) in each $(\mathrm{D}_s, \mathrm{D}_t)$:

$$\text{Transliteration similarity } (\mathrm{S},\mathrm{T},\theta) = \frac{1}{1+e^{-w^t\cdot\phi(\mathrm{S},\mathrm{T})}} \tag{2.6}$$

where $\phi(\mathrm{S},\mathrm{T})$ is the feature vector for the pair of words $\phi(\mathrm{S},\mathrm{T})$ and w is the weights vector which is learnt discriminatively using a bi-lingual list of matching transliterations. The second model, which they call the generative transliteration model extends a word alignment hidden Markov model (W-HMM) (He 2007). In a W-HMM, the emission probability depends on the current character and the previous target character. By marginalizing over all possible alignments, they compute the probability of a target word given a source word as follows:

$$P(\mathrm{T}|\mathrm{S}) = \sum_{A}\prod_{j=1}^{m} P(a_j|a_{j-1}, s_{a_{j-1}})P(t_j|s_{a_j}, t_{j-1}) \tag{2.7}$$

where t_j (and respectively s_i) denote the j^{th} (and respectively i^{th}) character in T (and respectively S) and $A \equiv a_1^m$ is the hidden alignment between T and S, and in which t_j is aligned to s_{a_j}, for $j = 1, ..., m$. The parameters of the W-HMM are estimated using the EM algorithm and they use $\log P(\mathrm{T}|\mathrm{S})$ for the transliteration similarity score.

Klementiev and Roth (2006) exploit two observations in mining transliteration pairs from multi-lingual news streams. The first observation is that NEs in one language associated with multi-lingual news streams tend to co-occur with their counterparts in another language for a given period of time. The second observation is that "NEs often contain or are entirely made up of words that are phonetically transliterated or have a common etymological origin across languages". Klementiev and Roth (2006) introduce an algorithm called *co-ranking* which exploits the two observations simultaneously during the transliteration detection process. For the first observation, they use a Discrete Fourier Transform (Arfken 1985) based metric to compute the similarity of time distributions. For the second observation, they score NE similarity using a discriminative linear transliteration model. The transliteration model is iteratively trained using single word NE pairs. During training, for a given source NE (S) in one language, the current model chooses a list of top-ranked

transliteration candidates T in another language. The words in the candidate pair are partitioned into a set of substrings su and tu up to a particular length (including the empty string which they denote by _). Couplings of the substrings (su, tu) from the source and target language sets of words produce feature vectors which are used for training. Klementiev and Roth employ the perceptron (Rosenblatt 1958) algorithm which takes a variable number of features in its examples; and as the iterative algorithm observes more data, it discovers and makes use of more features. Time sequence scoring is then used to rank the list and subsequently choose the target candidate NE *T'* that is best temporally aligned with *S*. A method called F-index (Hetland 2004) is used to implement the temporal similarity score function. The pairs of transliteration NEs and the best temporally aligned (thresholded) candidate NEs are utilized in a similar manner to iteratively train the transliteration model. The resulting pairs of source and target NEs are then evaluated for accuracy.

A recent shared task on transliteration mining (Kumaran et al. 2010b), used comparable Wikipedia article topics for five language pairs as source data for mining transliteration pairs. The comparable Wikipedia article topics are obtained with the aid of 'inter-language links' which are located on the same page as the source language text. A report about participating systems in the shared task shows the application of discriminative and generation-based methods for transliteration similarity estimation. Some approaches like those in Udupa et al. (2009) have been described above. Generation-based methods mostly included various forms of Hidden Markov Models (HMMs) and finite state automata (Darwish 2010, Noeman and Madkour 2010). Discriminative methods included some of the well known methods such as: support vector machines (SVMs) and a standard string kernel method (Jiampojamarn et al. 2010). The shared task report shows that some methods achieved good F-score values but on only one or a few datasets, and not for all datasets.

2.2.2 Discussion

The identification of candidate named entities can be a complex procedure depending on the requirements of the named entity recognition task and the language in use. As is suggested from some of the approaches reviewed above, some Asian language text presents more challenges for named entity recognition mainly because of the extra work required to identify words in a sentence as compared to text written using a Latin or Cyrillic alphabet. In Chinese text for example, the segmentation of a sentence is complicated by the lack of blanks and marks to indicate word boundaries. Consequently, the identification of candidate words is difficult with a major problem of segmentation ambiguities (Chen and Bai 1998).

The review above also shows that some methods specify the transliteration sim-

ilarity estimation problem in a similar manner but differ in the probabilistic models that contribute to a transliteration similarity estimate. For example, we see that a number of methods use the noisy channel model (NCM) approach but differ in the kinds of probability distributions that are specified to approximate the probability specifications in Equation 2.1 for the NCM. Other approaches apply an edit distance-based measure for example the Levenshtein Edit Distance based on a common representation for the source and target words. We also see variations in the computation of edit distance in different methods. Many generation-based approaches use a dynamic programming algorithm for evaluating alignments between the source and target words and use an Expectation Maximization algorithm for training the associated transliteration models. Up to this point, we see that although the HMM framework has been used in transliteration detection, there is not yet any application of the edit distance-based Pair HMM approach that we propose to use for the same task in the thesis. It is also clear from the review that there is not yet any application of generic Dynamic Bayesian Network (DBN) models such as the transduction-based DBN models proposed in the thesis for the same task. The review above also identifies the use of different discriminative-based approaches in transliteration detection.

The transliteration detection task has led to the organization of a shared task in which various transliteration detection methods are evaluated using the same standard corpora. Again as literature on transliteration detection generally revealed, both generative-based and discriminative-based methods were evaluated in the shared task and the report showed that there was still room for improvement. Such a shared task already simplifies some of our aims in the thesis. Specifically, by using the same datasets used in the shared task, we need not re-run the experiments associated with state-of-the-art approaches; we only have to run the experiments for the DBN models proposed in the thesis and compare the results with those that were reported for the state-of-the-art methods on the same standard transliteration corpora.

2.3 Transliteration generation

Unlike the transliteration detection task which can vary depending on the data source, the transliteration generation task (both *forward* and *backward*) is similar across different transliteration generation approaches. In *forward transliteration*, we want the transliteration system to automatically convert a source word into a target transliteration or a set of target transliterations when variations are expected. In the *backward*(reverse) direction, the aim is to find the original source word for a given target transliteration or transliterations. In both cases of transliteration generation, the core of the system is a trained model or set of models for character conversion. A

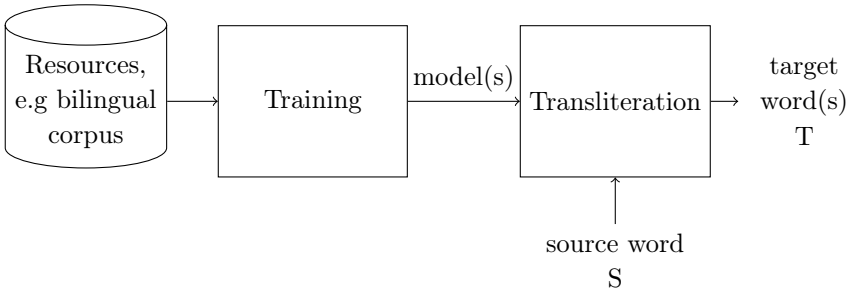

Figure 2.1: *Transliteration generation overview. Adapted from Karimi et al. (2011).*

general framework for transliteration generation is provided in (Karimi et al. 2011) specifying two main phases: training of the transliteration model(s) and generating transliterations using the trained model(s).

Karimi et al. (2011) identify different subtasks in each of the two main phases (Figure 2.1) in the transliteration generation process. Two common subtasks in the training phase include: segmentation of source and target character strings in the training pairs, and determining associations between the source and target transliteration units after performing a training procedure or a manual specification of the transformation rules. In the transliteration generation phase, two common subtasks include: source word segmentation into transliteration units, and using the trained model(s) or transformation rules to map source transliteration units to target transliteration units by resolving different combinations of alignments and unit mappings. As illustrated in Figure 2.1, at the end of the transliteration generation phase the transliteration system is usually expected to suggest more than one target transliteration. Since there is always a dependence on the previous sub-task from the training phase to the transliteration generation phase, we will review some of the main transliteration generation approaches in their entirety. Automated transliteration generation approaches are usually categorized according to the type of transliteration units used; that is, whether they are phonetic or orthographic or a combination of both (Karimi et al. 2011, Oh et al. 2006). In our review, we follow a similar categorization.

2.3.1 Phonetic-based transliteration generation

The earliest reported attempts at automated transliteration generation involved the use of phonemes between the source and target languages. In Arbabi et al. (1994), the generation of romanized transliterations for Arabic names is as follows: The Arabic names are first vowelized automatically using a combination of an artificial neural network (ANN) and a knowledge-based system (KBS). The ANN is used to

filter out words that would otherwise be vowelized inappropriately by the knowledge-based system; while the KBS uses linguistic vowelization rules. In the transliteration stage, the vowelized Arabic NEs can be converted into a standard, phonetic Latin representation using a parser or table. Generally, the Latin representation is broken down into a group of phonetic syllables which can be used to produce various spellings in languages that use the Latin alphabet.

A few years after Arbabi et al.'s (1994) work, Knight and Graehl (1997) used weighted finite-state automata for Japanese Katakana to English back transliteration. The transliteration process in Knight and Graehl (1997) follows a number of steps which are implemented as follows:

- English word (taken here as the source word S) sequences are generated using a distribution $P(\mathrm{S})$; then
- English pronunciations (SP) are produced from English words using $P(\mathrm{SP}|\mathrm{S})$;
- the English pronunciations are converted into Japanese sounds TP using $P(\mathrm{TP}|\mathrm{SP})$;
- Japanese sounds are converted into into Japanese Katakana (T_k) using $P(\mathrm{T}_k|\mathrm{TP})$;
- and misspellings caused by Optical Character Recognition (OCR) ($\mathrm{T}_{\mathrm{OCR}}$) are modeled through $P(\mathrm{T}_{\mathrm{OCR}}|\mathrm{T}_k)$.

$P(\mathrm{S})$ is implemented as a weighted finite state acceptor (WFSA), while the other conditional distributions are implemented as weighted finite state transducers (WFSTs). The aim is to find an English word S' that maximizes the joint probability given a Katakana string ($\mathrm{T}_{\mathrm{OCR}}$) observed by OCR:

$$\mathrm{S'} = \underset{\mathrm{S}}{\operatorname{argmax}}\, P(\mathrm{S}) \cdot P(\mathrm{SP}|\mathrm{S}) \cdot P(\mathrm{TP}|\mathrm{SP}) \cdot P(\mathrm{T}_k|\mathrm{TP}) \cdot P(\mathrm{T}_{\mathrm{OCR}}|\mathrm{T}_k) \tag{2.8}$$

Knight and Graehl (1997) use a general composition algorithm to integrate the different models and hence the computation in Equation 2.8. They then use Dijkstra's shortest-path algorithm (Dijkstra 1959) for extracting S'. A unigram scoring method is used in constructing the WFSA for $P(\mathrm{S})$; the WFST for $P(\mathrm{SP}|\mathrm{S})$ is based on the CMU pronunciation dictionary; the WFST for $P(\mathrm{TP}|\mathrm{SP})$ is learned automatically using an Estimation Maximization (EM) algorithm (Baum 1972) from a collection of English/Japanese sound sequences; the WFST for $P(\mathrm{T}_k|\mathrm{TP})$ is composed from two manually constructed WFSTs (the first WFST for merging long Japanese vowel sounds into new symbols while the second for mapping Japanese sounds to Katakana symbols); finally the WFST for $P(\mathrm{T}_{\mathrm{OCR}}|\mathrm{T}_k)$ is obtained using the EM algorithm applied on a collection of OCR'd text with the corresponding Japanese Katakana text.

Work related to Knight and Graehl (1997) then adapted and extended the phonetic-based approach while applying it to other language pairs. In Stalls and Knight (1998), weighted finite state automata are used in Arabic to English *back-transliteration*. Stalls and Knight (1998) use probability distributions similar to those in Knight and Graehl (1997) implementing the WFSA for P(S) and the WFST for P(SP|S) in exactly the same way. The modification in Stalls and Knight (1998) is that, instead of using conversions from English phonemes (SP) to Arabic phonemes (TP) and conversions from Arabic phonemes to Arabic orthography T, they use only one additional model for converting English phoneme sequences directly to Arabic orthography (P(T|SP). The WFST for P(T|SP) is obtained using the EM algorithm on a manually built English-phoneme-to-Arabic-writing dictionary. In Al-Onaizan and Knight (2002), the phonetic-based approach in Stalls and Knight (1998) is adapted by using a finite state machine to filter out ill-formed English phonetic sequences instead of using position markers in the phoneme set during the Arabic to English *back-transliteration* process. Al-Onaizan and Knight (2002) also extended Stall and Knight's (1998) phonetic-based approach with the use of a spelling based model to deal with words that are not of English origin.

Apart from the *back-transliteration* work based on Knight and Graehl's (1997) phonetic-based approach, there were other parallel phonetic-based approaches that employed different techniques for transliteration generation in different language pairs:

Kawtrakul et al. (1998) performed Thai-to-English *back-transliteration*. In Kawtrakul et al. (1998), Thai loan words (T) were first segmented into syllables and mapped to phonemes using some transcription rules. The phoneme sequences of the loan words were then compared to the phonetic sequence of a set of English words (S) in a phonetic dictionary. The English word (S') with the most similar phonetic sequence was selected as the transliteration.

Jung et al. (2000) applied an *extended Markov window* method to build the model for English to Korean transliteration. In their transliteration process, Jung et al. (2000) first generate mappings between English and Korean phonemes. They use the pronunciation symbols for English words as defined in the Oxford computer-usable dictionary (Roger 1992) and construct English to Korean phonetic mapping tables that meet syllabification and alignment requirements for training the transliteration model. Their alignment process proceeds in two stages: consonant alignment obtained from a scan of English phonetic units and Korean notation; and vowel alignment which leads to a separation of corresponding vowel pairs based on the consonant alignment stage. In the transliteration generation stage, a probabilistic tagger is used to find the most likely Korean transliteration candidates given an English input NE that has been syllabified. Given that S represents an English NE (where

Ss = $ss_1 ss_2 ... ss_n$ is its syllabification), and T = $tp_1 tp_2 ... tp_n$ a Korean word (where tp_i is the i^{th} phonetic unit of T), Jung et al. (2000) aim at finding a Korean word T' such that the joint probability $p(S,T) \cong p(Ss,T)$ is maximized.

$$\text{T'} = \underset{\text{T}}{\operatorname{argmax}} P(\text{Ss,T}) = \underset{\text{T}}{\operatorname{argmax}} P(\text{T}|\text{S})P(\text{S}) \tag{2.9}$$

where the translation model in Equation 2.9 is approximated based on the extended Markov window as follows:

$$P(\text{T}|\text{S}) \cong \prod_i P(tp_i|tp_{i-1}, ss_{i-1}, ss_i, ss_{i+1}) \tag{2.10}$$

Equation 2.10 is further expanded into more fragmented probability terms to deal with data sparseness when training. For the language model probability (P(Ss)) in Equation 2.9, a bi-gram language model is used:

$$P(\text{Ss}) \cong \prod_i P(ss_i|ss_{i-1}) \tag{2.11}$$

the transliteration model is then finally formulated as:

$$\text{T'} = \underset{\text{T}}{\operatorname{argmax}} P(\text{Ss,T}) \cong \underset{\text{T}}{\operatorname{argmax}} \prod_i \frac{P(tp_i|ss_{i-1}, tp_{i-1})P(ss_i|tp_i, ss_{i-1})P(ss_{i+1}|tp_i, ss_i)}{P(ss_{i+1}|ss_i)} \tag{2.12}$$

Lee and Choi (1998) model English to Korean transliteration by using both phoneme transformations (*pivot method* corresponding to phonetic-based transliteration), and only grapheme transformations (*direct method*). In their phoneme-based transliteration method, English graphemes are first converted to English phonemes; the English phonemes are then converted to Korean graphemes. Using S to denote an English word and T a Korean word, the aim is to find a Korean word T' that maximizes the conditional probability P(T|S). By applying Bayes' rule, Lee and Choi specify their transliteration problem as:

$$\text{T'} = \underset{\text{T}}{\operatorname{argmax}} P(\text{T}|\text{S}) = \underset{\text{T}}{\operatorname{argmax}} \frac{P(\text{T})P(\text{S}|\text{T})}{P(\text{S})} \cong \underset{\text{T}}{\operatorname{argmax}} P(\text{T})P(\text{S}|\text{T}) \tag{2.13}$$

The language model probability P(T) in Equation 2.13 is obtained by using a bi-gram conditional probability distribution on Korean pronunciation units (PUs):

$$P(\text{T}) \approx \sum_{i=1}^{\text{N}} \prod^{t} P(tp_i|tp_{i-1})$$

where N is the total number of segmentation of T, tp_i is the i^{th} PU in a segmentation of T, and t is the total number of PUs in a segmentation. The translation

model probability $P(\mathrm{S}|\mathrm{T})$ is also obtained using a bi-gram conditional probability distribution between English and Korean PUs:

$$P(\mathrm{S}|\mathrm{T}) \approx \sum^{\delta} \prod_{i=1}^{t} P(sp_i|tp_i) \text{ where } \delta \text{ is the total number of alignments for T and S.}$$

Wan and Verspoor (1998) use five stages in English to Chinese transliteration. in the first stage they parse a complete English phrase through a dictionary in search of a standard translation, and if there is no standard translation, the phrase is broken into words, and each word is parsed through a dictionary. Words with no standard translations are selected for transliteration. In the second stage, each word that is selected for transliteration is divided into syllables, and in the third stage the transliteration process proceeds to find patterns within each syllable that are handled in appropriate ways for mapping to a particular Romanization standard (Pinyin in this case) in the fourth stage. In the last stage, the Pinyin representation of a word is mapped to Chinese Han characters using a Pinyin - Han character correspondence table.

Meng et al. (2001) use a number of modules based on the steps required to transform an English out of vocabulary word to Chinese. Meng et al. (2001) first detect Romanized Chinese names using a maximum-matching segmentation algorithm and then automatically acquire pronunciations for names other than Romanized Chinese names using either the PRONLEX pronunciation lexicon from LDC or an automatic letter-to-phoneme generation process (which is obtained through training on the PRONLEX lexicon by aligning words with the corresponding pronunciations in Viterbi-style for a one-to-one letter-to-phoneme mapping). Meng et al. then apply cross-lingual phonological rules to deal with some problems in English pronunciations. For phoneme alignments between English and Chinese, Meng et al. iteratively train a finite state transducer using a bi-lingual proper name list containing English names and their Chinese transliterations. Given an English phoneme sequence, they implement confusion matrices to produce alternative Chinese phoneme sequences prior to syllabification in a Chinese Romanization system (Pinyin in this case). The result is called a Chinese phoneme lattice. In the final stage, they search through the phoneme lattice to identify Chinese phonemes that constitute legitimate syllables. The resulting syllable graph is searched using the A* search algorithm to find the N most probable syllable sequence using probabilities derived from the confusion matrix and a syllable bi-gram language model.

Karimi et al. (2011) review a number of additional phonetic-based transliteration generation approaches. To avoid a complete repetition of the review in Karimi et al. (2011), we provide a summary of some of these other phonetic-based approaches while focusing on the transliteration models used.

Jeong et al. (1999) use a Hidden Markov Model (HMM) framework in Korean to English *back-transliteration*. The aim is to determine the most likely original English (*S'*) word that maximizes the conditional probability of an English word given a "foreign" word ($P(\mathrm{S}|\mathrm{T})$). Using $\mathrm{S} = s_1 s_2 ... s_m$ to denote the English word, and $\mathrm{T} = t_1 t_2 ... t_n$ to denote the foreign word, Jeong et al. (1999) formulate the transliteration problem as follows:

$$\begin{aligned} S' &= \underset{\mathrm{S}}{\operatorname{argmax}} P(\mathrm{S}|\mathrm{T}) = \underset{\mathrm{S}}{\operatorname{argmax}} P(s_1 s_2 ... s_m | t_1 t_2 ... t_n) \\ &= \underset{\mathrm{S}}{\operatorname{argmax}} P(s_1 s_2 ... s_m) \times P(t_1 t_2 ... t_n | s_1 s_2 ... s_m) \end{aligned} \tag{2.14}$$

Using a phonetic representation ($sp_1 sp_2 ... sp_p$) for the English string $\mathrm{S} = s_1 s_2 ... s_m$, Equation 2.14 leads to:

$$\begin{aligned} \mathrm{S'} &= \underset{\mathrm{S}}{\operatorname{argmax}} P(sp_1 sp_2 ... sp_p) \times P(t_1 t_2 ... t_n | sp_1 sp_2 ... sp_p) \\ &\cong \underset{\mathrm{S}}{\operatorname{argmax}} \prod_j P(sp_j | sp_{j-1}) \times P(t_j | sp_j) \end{aligned} \tag{2.15}$$

The first term in Equation 2.15 $P(sp_j|sp_{j-1})$ corresponds to the transition probability between two states of an HMM while the second term $P(t_j|sp_j)$ to the output probability in a given state. The computation in Equation 2.15 is effected using the Viterbi algorithm.

Oh and Choi (2002) use pronunciation and contextual rules for English to Korean transliteration. In Oh and Choi's system, English pronunciation units are first aligned to corresponding phonemes, then the transliteration of English words to Korean words is achieved through a number of steps including: identification and processing of "complex word forms"; detection and processing of English words of Greek origin; chunking of aligned English pronunciation and phoneme data into two classes of *pure English words* and *English words of Greek origin*); and finally English phoneme to Korean conversion based on the use of English to Korean Standard Conversion Rule (EKSCR). Contextual rules are captured by observing errors from the use of EKSCR to a given number of randomly selected words which are not part of the test set.

Virga and Khudanpur (2003) apply the IBM source-channel model (Brown et al. 1993) in English to Chinese transliteration. The steps followed in their transliteration process include: 1) conversion of an English name into a phonemic representation using the Festival speech synthesis system; 2) translation of the English phoneme sequence into a sequence of generalized initials and finals (GIFs) which are the commonly used sub-syllabic units for expressing pronunciations of Chinese characters; 3) transformation of GIF sequences into Pinyin symbols without specifying tone; and

4) translation of the Pinyin sequence to Chinese character sequence. Steps 2 and 4 are accomplished using Brown et al's (1993) statistical translation model. For example, for step 2, the aim is to find the sequence of GIF symbols $g' = g'_1 g'_2 ... g'_k$ that maximize the probability of a GIF sequence $g = g_1 g_2 ... g_l$ given an English phoneme sequence $SP = sp_1 sp_2 ... sp_j$:

$$g' = \underset{g}{\operatorname{argmax}} P(g|sp) = \underset{g}{\operatorname{argmax}} P(sp|g)P(g). \tag{2.16}$$

Virga and Khudanpur (2003) estimate a trigram model for $P(g)$ in Equation 2.16 using a CMU toolkit on a training portion of Chinese names. In the case of step 4, a trigram model with Good-Turing discounting and Katz back-off is estimated as the language model for the transformation of Pinyin sequences to Chinese characters.

Gao et al. (2005) used a *direct model* in English to Chinese transliteration as opposed to the *indirect* approach in the source channel model described above (Virga and Khudanpur 2003). Using TC $= tc_1 tc_2 ... tc_l$ to represent Chinese Pinyin sequences and SP $= sp_1 sp_2 ... sp_j$ for the English phoneme sequence as above, Gao et al. (2005) reformulate the transliteration problem by rewriting Equation 2.16:

$$\text{TC'} = \underset{\text{TC}}{\operatorname{argmax}} P(\text{TC}|\text{SP})P(\text{TC}) \tag{2.17}$$

Gao et al. (2005) use an EM algorithm to find the Viterbi alignment per training pair for generating English phoneme to Chinese Pinyin mapping probabilities, which are subsequently encoded in a WFST. For the language model for P(TC), they train a syllable-based bi-gram model using the same instances of Chinese names that were used for building the WFST.

Most of the reviews in literature about phonetic-based machine transliteration hardly discuss any work of the kind in the last five years. Below we briefly review only two recent references where phonetic-based transliteration is used. In both references, an Indian language is involved in the phonetic-based transliteration process. While phonetic-based transliteration is used in both references, orthographic-based transliteration is also reported and seems to be used more than the phonetic-based approach.

Surana and Singh (2008) use pronunciations for foreign words in English to Indian language transliteration. They use the the CMU speech dictionary for lookup and for training a pronunciation estimation model. English words that are not of Indian origin are first converted to phonemic representation and the English phonemes are then mapped to Indian language letters. For English words that are of Indian origin, they simply segment the English word and convert the Latin segments into Indian language segments for generating a corresponding Indian transliteration.

Das et al. (2010) use a phonetic-based transliteration approach to handle valid English dictionary words. They use a standard machine learning sequence labeler conditional random field to map English phonemes to Indian language transliteration units.

2.3.2 Orthographic-based transliteration generation

Going by the common definition for transliteration generation, the transliteration process is expected to involve a phonetic mapping from one language to another. However, transliteration work as reported in recent literature suggests that, orthographic-only based methods result in transliteration generation quality which is comparable to that for phoneme-based methods, and sometimes even significantly better (Li et al. 2004). An orthographic-based approach eliminates a number of intermediate phonetic representation and transformation steps that require extra work and time. The elimination of the intermediate phonetic steps implies that any flaws that may be associated with the intermediate steps will be avoided. Recent work has mostly favored the use of orthographic-based transliteration generation, mainly applying techniques associated with the machine learning paradigm. Most of the earlier orthographic-based methods have been reviewed well in recent reviews on machine transliteration (Oh et al. 2006, Chinnakotla et al. 2010, Karimi et al. 2011). The orthographic-based approach constitutes various types of approaches with the two notable categorizations of generative-based and discriminative-based models as is the case for transliteration detection above. Although we discuss some of the transliteration generation methods under the two common categorizations of generative-based and discriminative-based approaches, we also separately discuss recent approaches that are adapted from a related domain (especially machine translation (Matthews 2007, Finch and Sumita 2008)), and those that use additional information in the transliteration generation process such as semantic transliteration by Li et al. (2007).

a) Generative and Rule-based approaches

The earliest cited work on orthographic-based transliteration generation suggests a common usage of a machine translation inspired framework of the source channel model (Lee and Choi 1998, Jeong et al. 1999, Kim et al. 1999) and related methods such as the joint source channel model (Li et al. 2004, Zhang et al. 2004) and modified joint source channel model (Ekbal et al. 2006). Other earlier transliteration generation modeling methods include: decision trees (Kang and Choi 2000), transliteration networks (Kang and Kim 2000, Goto et al. 2003), and n-gram models (Abduljaleel and Larkey 2003). All of these are generation-based approaches.

Relatively recent approaches include: Weighted Finite State Transducers (WFSTs) (Lindén 2006), hand crafted transliteration rules (Malik 2006), consonant-vowel-based methods (Karimi et al. 2006, Karimi et al. 2007), and substring-based transduction (Sherif and Kondrak 2007b). In the following we point out the main techniques that are employed starting with earlier approaches to recent approaches.

Kang and Choi (2000) used decision trees to generate Korean strings given English words. They use an extended version of Covington's (1996) alignment algorithm to determine alignments for training the decision trees. In the extended version of Covington's alignment algorithm, they introduce a binding operation to deal with 'null mappings'. A *depth-first search* algorithm is used to prune away fruitless branches when estimating the alignments. To learn decision trees using the alignments, ID3 like algorithms (Quinlan 1986) are used. During transliteration, each English letter in a given English word is mapped to Korean characters using the corresponding decision trees; the Korean characters are then concatenated to produce the final Korean transliteration.

Goto et al. (2003) use several models based on a lattice of conversion units between English and Japanese Katakana characters. During transliteration, Goto et al.'s method follows three approaches: the computation of the likelihood of a particular choice of generating English conversion units through letter chunking for a given English word; the use of English and Japanese contextual information simultaneously to compute the plausibility of conversion from each English conversion unit to various Japanese conversion candidate units using a single appropriate probability model; the use of several probability models based on the *maximum entropy method* while modeling different kinds of information.

Abduljaleel and Larkey (2003) use an n-gram transliteration model for English to Arabic transliteration. The model is a set of conditional probability distributions over Arabic characters, conditioned on English unigrams and selected n-grams. They use proper name lists to train the n-gram model using GIZA++ (a statistical alignment tool). During transliteration, an English word S is first segmented according to the n-gram inventory, and for each segment, all possible Arabic transliterations T are generated. The equation for scoring each word is given as follows:

$$P(\mathrm{T}|\mathrm{S}, \mathrm{T} \in \mathrm{Ar}) = P(\mathrm{T}|\mathrm{S}) \times P(\mathrm{T} \in \mathrm{Ar}) \tag{2.18}$$

where $P(\mathrm{T} \in \mathrm{Ar})$ is the probability that the Arabic word T conforms to the spelling patterns of Arabic names, and is computed using a letter bigram model of general Arabic as the product of the probabilities of each letter bigram in T.

Li et al. (2004) and Zhang et al. (2004) use a joint source channel model to capture the simultaneous generation of source and target words. A joint probability model is estimated and is marginalized to yield conditional probability models for

both *forward transliteration* and *back-transliteration*. Given an alignment δ with transliteration unit correspondences $\langle s,t\rangle_k$ for an English string S and a Chinese string T, Li et al. formulate their transliteration problem as follows:

$$\text{T'} = \underset{\text{T},\delta}{\operatorname{argmax}} P(\text{S},\text{T},\delta) \text{ for English to Chinese transliteration, and}$$

$$\text{S'} = \underset{\text{S},\delta}{\operatorname{argmax}} P(\text{S},\text{T},\delta) \text{ for Chinese to English transliteration.}$$

Li et al. (2004) then use an n-gram transliteration model to capture the conditional probability or transliteration probability of a transliteration unit correspondence $\langle s,t\rangle_k$ depending on its immediate n predecessors. The following Equation is used to compute the joint probability of a pair of English (S) and Chinese (T) words:

$$P(\text{S},\text{T}) = P(\text{S},\ \text{T},\delta) = \prod_{k=1}^{K} P(\langle s,t\rangle_k | \langle s,t\rangle_{k-n+1}^{k-1}) \tag{2.19}$$

Lindén (2006) aligns source and target words using the minimum simple edit distance. From the alignments, Lindén determines the frequency of each edit operation in context for at most four letters in the source word including the letter s_i which is aligned with t_i in the target word T. Given that the context $s_{i-1}, s_i, s_{i+1}, s_{i+2}$ in the source word is represented by s_{i4}, Lindén formulates the transliteration problem as follows:

$$P(\text{T}|\text{S}) = \prod_{i=1..\max(|\text{T}|,|\text{S}|)} P(t_i|s_{i4}) \tag{2.20}$$

$P(t_i|s_{i4})$ is estimated with counts of the transformations $t_i|s_{i4}$ divided by the count of the context s_{i4}. When the context rarely occurs, an offline back-off model is used for smoothing $P(t_i|s_{i4})$. Lindén then uses a cascade of weighted finite state transducers to implement the transliteration process.

Malik (Malik 2006) employs a completely rule-based approach for transliteration from *Shahmukhi* to *Gurmukhi*. During transliteration, each *Shahmukhi* token is parsed into its constituent characters. Characters that bear a dependency are transliterated using 'dependency rules' while characters that do not bear a dependency are transliterated by character mapping. Malik (Malik 2006) specifies a number of tables that encode the different types of rules.

Karimi et al. (2007) use an alignment approach comprised of two steps: the first step uses consonant and vowel properties of a word's characters, and the second uses a frequency-based search for valid alignments. In the first step, consonant-vowel sequences (q_S and q_T) for a pair of words (S and T) in a training corpus is

generated and if the sequences match, consonant clusters and vowel sequences are added to an alignment set. if q_{S} and q_{T} do not match, the second step is used where a search for alignments proceeds from left to right while examining one of four possible options for transliteration: single character to single character (s_i, t_j, r), digraph to single character (s_i, s_{i+1}, t_j, r), single character to digraph (s_i, t_j, t_{j+1}, r), and single character to empty string (s_i, ϵ, r). For transliteration generation, Karimi et al. (2007) propose the use of a *collapsed consonant and vowel* method called (CV-MODEL3) as an extension of two previous models (CV-MODEL1 and CVMODEL2). The source word is segmented and a probability is computed for each generated word (T) as follows:

$$P(\mathrm{T}|\mathrm{S}) = \prod_{k=1}^{|K|} P(\hat{\mathrm{T}}_k|\hat{\mathrm{S}}_k),$$

where $|K|$ is the number of distinct source segments and $P(\hat{\mathrm{T}}_k|\hat{\mathrm{S}}_k)$ is the probability of the $\hat{\mathrm{S}}_k \rightarrow \hat{\mathrm{T}}_k$ transformation rule. Karimi et al then apply a tree structure following Dijkstra's α-shortest path, to generate the α highest scoring (most probable) transliterations, ranked based on their probabilities.

b) Semantic transliteration

Li et al. (2007) introduce into the transliteration model semantic information with regard to language of origin and the gender associated with a name. The aim is still to determine the optimum target name *T'* which yields the highest posterior probability given the source name S:

$$\mathrm{T'} = \underset{\mathrm{T}\in\tau_{\mathrm{S}}}{\operatorname{argmax}} P(\mathrm{T}|\mathrm{S}) \tag{2.21}$$

where τ_S is the set of all possible transliterations for the source name S. To incorporate language of origin (L) and gender information (G) in the transliteration, Equation 2.21 is re-written as:

$$P(\mathrm{T}|\mathrm{S}) = \sum_{\mathrm{L}\in\mathcal{L},\mathrm{G}\in\mathcal{G}} P(\mathrm{T},\mathrm{L},\mathrm{G}|\mathrm{S}) = \sum_{\mathrm{L}\in\mathcal{L},\mathrm{G}\in\mathcal{G}} P(\mathrm{T}|\mathrm{S},\mathrm{L},\mathrm{G})P(\mathrm{L},\mathrm{G}|\mathrm{S}) \tag{2.22}$$

where P(T|S,L,G) is the transliteration probability from source S to target T, given the language of origin L and gender G. $\mathcal{L}$ and $\mathcal{G}$ denote the sets of languages and gender respectively. Given an alignment between S and T, P(T|S,L,G) is estimated using a bigram language model. The mappings between source and target characters for computing P(T|S,L,G) are obtained from alignments resulting from the application of the EM algorithm on training data. Information concerning gender

and language of origin is incorporated in Equation 2.22 by rewriting P(L,G|S) as P(L,G|S) = P(G|L,S)P(L|S). Using L_S to denote the language of S, P(L|S) can be obtained as:

$$P(\mathrm{L}|\mathrm{S}) = \begin{cases} 1 & \mathrm{L} = \mathrm{L_S} \\ 0 & \mathrm{L} \neq \mathrm{L_S} \end{cases}$$

and using G_S to denote the gender for S, $p(G|L,S)$ is obtained as:

$$P(\mathrm{G}|\mathrm{L},\mathrm{S}) = \begin{cases} 1 & \mathrm{G} = \mathrm{G_S} \\ 0 & \mathrm{G} \neq \mathrm{G_S} \end{cases}$$

In the case where semantic information is not available, Li et al. (2007) learn the semantic information from the names themselves.

c) Transliteration using statistical machine translation methods

While a number of approaches aim to develop transliteration-specific methods, the adaptation of Statistical Machine Translation (SMT) methods to transliteration generation has become popular as a valuable alternative. The SMT methods that have been used for transliteration generation range from the earlier popular IBM models (Brown et al. 1993) to a currently more popular state-of-the-art phrase-based SMT approach (Koehn et al. 2003). The application of the SMT methods is simply modified to reflect the properties of a transliteration process. That is, the task is first viewed as a character translation problem rather than a word (or phrase-based translation) problem. In section 2.3.1 on *phonetic-based methods for transliteration generation*, we have already seen the application of the IBM models (Virga and Khudanpur 2003). Orthographic-based methods that use the IBM models, utilize them in a manner similar to that presented for the phonetic-based approaches. A detailed description about the adaptation of the phrase-based SMT approach to transliteration can be found in Matthews (2007) and Finch and Sumita (2008).

d) Discriminative machine transliteration

Zelenko and Aone (2006) propose two discriminative methods for transliteration. Using an existing transliteration dictionary D (a set of name pairs (S,T)), Zelenko and Aone learn a function that directly maps a name S from one language into a name T in another language. The main difference in their work is the omission of the alignment step and any probabilistic computations such as P(T|S), P(S, T) that depend on alignments. Their discriminative methods correspond to local and global modeling paradigms: in the local paradigm, Zelenko and Aone learn linear classifiers that predict a letter t_i from the previously predicted letters $s_1 ... s_{i-1}$ and the original

name *S*. In the global paradigm, Zelenko and Aone learn a function W for mapping a pair (S,T) into a score $W(S,T) \in \Re$.

Klementiev and Roth (2006) train a linear model to decide whether a target word (T) from a set of candidate words is a transliteration of a source word (S). T and S are partitioned into a set of substrings Ss and Ts up to a particular length (including the empty string). Klementiev and Roth use the same transliteration model described in the previous section on *Transliteration Detection* to convert input strings to target strings.

e) NEWS 2009 and NEWS 2010 shared tasks on transliteration generation

Just like the case for transliteration detection, the transliteration generation task had also led to the organization of two transliteration generation shared tasks (Li et al. 2009, Li et al. 2010) to evaluate state-of-the-art transliteration generation methods using the same standard corpora by the time of writing this thesis. The reports from the two shared tasks (Li et al. 2009, Li et al. 2010) also show the use of both generation-based and discriminative-based methods for modeling transliteration generation.

The 2009 NEWS shared task report (Li et al. 2009) identified two transliteration generation modeling approaches that were applied by many of the participating teams: phrase-based statistical machine transliteration (which originates from statistical machine translation work as described above) and Conditional Random Fields (CRFs) (Lafferty et al. 2001). The most successful approaches in the first shared task, however, are reported to have combined several models (CRFs, Maximum Entropy Models, Margin Infused Relaxed algorithm) by re-ranking the transliteration generation outputs from each model (Oh et al. 2009). A discriminative sequence prediction model (Jiampojamarn et al. 2009) referred to as DirectL was reported to have a good transliteration generation performance.

The 2010 NEWS shared task report (Li et al. 2010) shows reduced participation, but for the teams that participated, approaches that are similar to those applied in the 2009 shared task are used including phrase-based machine transliteration and CRFs. The phrase-based approach is used for transliteration on various language pairs (Finch and Sumita 2010, Song et al. 2010) while CRFs are used by one of the seven participating teams (Das et al. 2010). Jiampojamarn et al. (2010) extend their DirectL approach above resulting in relatively better transliteration generation quality for this shared task. Most of the participating approaches also combine different models via re-ranking of the outputs (Das et al. 2010, Finch and Sumita 2010, Song et al. 2010) to improve transliteration generation quality. All methods are said to be orthographic-based except for some cases where a Romanization system is used before applying a transliteration generation system. However, in almost all

results associated with the twelve language pairs for the 2010 shared task, only one participating system achieves just over 50 % transliteration generation *word accuracy* on the English-Korean language pair.

2.3.3 Discussion

Approaches for automated transliteration generation can be divided into two main classes: those that use phonetic information in the transliteration process and those that use only the orthographic representation. Transliteration generation literature shows that phonetic-based approaches were mainly used during the early years of research in automating the transliteration generation process. Later and more recently, orthographic-based approaches seem to be more preferred. However, there are also techniques that combine the use of both phonetic and orthographic-only information.

As the review shows, most of the approaches aim at determining a target word (in the case of forward transliteration) that maximizes a posterior probability associated with the target word given the source. Again, we see that the noisy channel framework is used for transliteration generation as well and the channel (transliteration) model is specified and implemented differently in the reviewed approaches. Although there are some approaches that use an alignment-based representation to model the transliteration generation process, explicit references to the use of an edit distance-based metric to score hypothetical transliterations are rare. The review also shows that the HMM framework has been used mainly for learning alignments for transliteration generation. This includes being part of the alignment models in the GIZA++ alignment toolkit which is used when applying state-of-the-art statistical machine translation approaches to transliteration generation. Here, we envisage that the DBN models proposed in the thesis can be applied in a manner similar to how the HMMs have been used in learning alignments for transliteration generation . However, there has not yet been an application of the DBN models that we propose for that purpose.

We also see that, some approaches incorporate linguistic information to help improve transliteration generation quality. Recently, there are attempts to incorporate different types of information such as semantic information in addition to using the orthographic-only information (Li et al. 2007). The review also shows that improved transliteration generation quality can be achieved by combining the application of transliteration generation methods compared to applying each method separately as the reports from the recent shared tasks on transliteration generation show (Li et al. 2009, Li et al. 2010). However, results associated with the 2010 shared task suggest that there is still considerable room for improving transliteration generation quality since almost none of the state-of-the-art systems achieved over 50% transliter-

ation generation accuracy using standard transliteration corpora for twelve language pairs. Evaluation of the state-of-the-art systems in the 2011 shared task on transliteration generation was still onging at the time of writing this thesis.

2.4 Conclusion

Literature on automated transliteration detection and generation suggests generic constituent phases for which various methods have been proposed to model transliteration related tasks. For transliteration generation, many approaches follow an overall two step procedure of first specifying or training transliteration models given correct transliteration pairs, and then later applying the models to propose target candidate words given a source language word. The transliteration generation process also usually involves first specifying or training a transliteration model to be used for computing transliteration similarity in the identification of transliteration pairs from a pre-processed set of candidate transliterations. It is often the case that the size of training data affects the representational quality and consequently the performance of the transliteration models.

Recent shared tasks on transliteration mining (Kumaran et al. 2010b) and generation (Li et al. 2009, Li et al. 2010) provide us with some baseline for making general conclusions about the current state of research. In both shared tasks most of the approaches are reported to as using similar techniques with very few modifications. For example, the reports for all the previous shared tasks on transliteration generation show that a large percentage of the systems used either a phrase-based statistical machine translation approach or CRFs. Although some some of these approaches achieved good performances on some language pairs, the shared task results show that they are still far from achieving high transliteration detection and generation quality for many other language pairs.

It is clear from our literature review that most of the work on machine transliteration has been concentrated on cases where the source and target languages use different writing systems. In this thesis, we propose the application of the traditional transliteration detection and generation setups for cases where source and target use the same writing system. We have based our proposition on the fact that certain differences between languages that use the same writing system result in different written representations for the same entity. And that this is similar to how different representations are used for the same entity across writing systems. Therefore traditional transliteration setups should be a plausible alternative to dealing with *unknown* words across languages that use the same writing system.

Finally, although literature shows that the HMM framework has been applied in both transliteration detection and generation, there are many other HMM and

DBN model generalizations that have not yet been evaluated in the two tasks. This includes the Pair HMM and transduction-based DBN models that we propose to apply in the two tasks. In the following chapter, we provide an overview on the concepts underlying the framework of Dynamic Bayesian Networks before exploring in later chapters our application of the proposed edit distance-based DBN modeling approaches in transliteration detection and generation.

Chapter 3

Dynamic Bayesian Networks

3.1 Introduction

Dynamic Bayesian networks (DBNs) are a class of temporal probabilistic graphical models (PGMs) that have found successful application in many domains. This is attributed to the more general representations that DBNs allow, leading to very large model spaces and the use of generic algorithms for inference and learning. The DBN framework already generalizes a variety of methods including some of the most common and successful methods in Natural Language Processing (NLP) such as Hidden Markov Models (HMMs). The inference and learning algorithms used in these methods can also be viewed as instantiations of some of the standard DBN algorithms; for example, the forward-backward algorithm used for inference and learning in HMMs can be considered a special type of the message-passing algorithm used for inference in Bayesian Networks.[1]

As temporal probabilistic graphical models, DBNs are used to model not only sequential data (linguistic or biological) where we are doubtful about the generating mechanism but also to model time series data that is generated by some causal process (Murphy 2002). Various DBN modeling methods have been proposed for many tasks in the literature, and most of the methods are easily adaptable to various other tasks. Although, it should be important to review the various DBN modeling approaches when proposing to apply DBN models in a given task, we do not present such a review in this thesis. The reader is referred to Murphy (Murphy 2002) for some examples of DBN modeling approaches. In this chapter, we present abstractly the concepts underlying the framework of Dynamic Bayesian Networks in three aspects: the specification and representation of DBN models, DBN inference, and DBN

[1] We use the term Dynamic Bayesian Networks to generalize HMMs regardless of how the HMMs are implemented.

learning. We as well propose some examples to illustrate the application of some of the concepts in the context of transliteration. In the following section, we start with an introduction to Bayesian networks from which DBNs are an extension.

3.2 Bayesian networks

3.2.1 Representation

Bayesian networks provide a means of expressing a joint probability distribution over a set of inter-related random variables. They are specified by way of a graphical modeling language where nodes are used to represent random variables and edges are used to represent dependencies between the random variables for some system domain. The use of a graphical language to communicate representative models carries with it a number of advantages. Some advantages given by Koller and Friedmann (2009) are as follows: 1) A graphical representation provides an accurate reflection of our understanding of the domain we are modeling and facilitates the effective construction of the models. 2) A graphical representation allows the distributions defined by a given model to be used effectively for *inference* where there is need to answer different types of queries with respect to the problem domain.

In specific terms, Bayesian Networks are described as *directed acyclic graphs* in which edges specify conditional dependencies or independencies. The graphical representation of a network also specifies the requirements for the quantitative part of the model which comprises of a set of probability distribution functions for each random variable. A formal definition of a Bayesian network specifies the following (Jensen and Nielsen 2007, Koller and Friedman 2009):

- A set of random variables $X_1, ..., X_n$ represented by nodes and a set of directed edges between the random variables.
- Each variable has a finite set of mutually exclusive states.
- The variables together with the directed edges form a directed acyclic graph (DAG).
- Each variable X_i is conditionally dependent on only its parents (Pa_{X_i}) and its children.
- To each variable X_i with parents Pa_{X_i}, a conditional probability table (CPT) for $P(X_i|\mathrm{Pa}_{X_i})$ is attached.

As suggested by the definition above, a number of stages are involved in constructing a Bayesian network model to represent a given domain. Koller and Friedman (2009)

introduce three important stages to Bayesian network model construction which we review based on their presentation in the following section.

a) Specifying variables

In a Bayesian network, variables are used to represent entities that are relevant to the domain we are modeling. The entities and their related attributes may be described in various ways, but it is important that the variables that we use do precisely represent the domain. If the variables do not precisely represent the domain, conclusions resulting from the use of the Bayesian network model will be inaccurate for a given set of observations in the domain. In addition to using variables that precisely represent a problem domain, we also have to ensure that each variable's range of values adequately represent the true conditional independence assumptions as defined in a Bayesian network model of the domain.

Based on how the problem has been specified, different types of variables can be defined including *observable* and / or *hidden* variables. Observable variables are variables that we can directly measure whereas hidden variables are variables that we can only infer from the observable variables in the model. Also, based on the problem, the variables may be *discrete* or *continuous*. In modeling transliteration, we expect to use only discrete random variables, that is variables that take on a finite set of values.

b) Specifying the Bayesian network structure

Although a Bayesian network is used to compactly represent the joint probability distributions for a particular domain, the specification of a Bayesian network structure is not straightforward. This is because there can be many network structures that reflect the same set of independencies in the domain we are modeling. Koller and Friedman (2009) suggest an approach that should be successful for specifying a Bayesian network to represent the domain of interest. This approach according to Koller and Friedman is to specify a structure that in the most part reflects the causal order and dependencies of the variables in the domain, so that between two variables, we use the notion 'child variable' to represent the *effect* and the notion *parent variable* to represent the *cause*. Koller and Friedman (2009) also point out a commonly used approach of using a *backward construction process* in the specification of a Bayesian network structure. The process is called *backward* because when constructing the Bayesian network, we start with a child variable which represents an effect and move on to determine factors (causes) that are associated with the effect which we add to the network as parent variables having edges to the child variable. Koller and Friedman (2009) also emphasize the inevitability of approximations where

it is possible to represent many weak dependencies which can easily result in a very complex model that is infeasible to use.

c) Specifying probabilities

This is a very important task in completing the specification of a Bayesian network model since the queries that we would like to answer about the domain rely on probability distributions encoded by the Bayesian network. Koller and Friedman (2009) point out a number of errors that can have a significant effect on the conclusions that result from using a Bayesian network. Examples of errors from Koller and Friedmann (2009) that we need to handle when assigning probability distributions for different variables in the network include: *zero probabilities* which need to be avoided in appropriate ways; the presence of *small differences* in probabilities that could imply large differences in conclusions; and the insensitivity of the model to differences in relative probabilities, for example we would require that the Bayesian network model encode correctly that the probability of the relationship between an English l and a Russian л is greater than the relationship between an English l and the Russian ж.

Practically, the probabilities can be assigned through an expert or can be estimated from training examples of the domain. In the context of transliteration, some of the variables used may be associated with large probability distributions that we can not manually assign but can only use a computational technique for automatic estimation. A common probability estimation approach that we review later in this chapter is the Expectation Maximization procedure.

3.2.2 A transliteration example

We would like to represent the character relationships between writing systems used by some source and target languages (for example English and Russian respectively). We can use a random variable s_i to represent a character in the source language writing system, and a random variable t_j to represent a character in the target language writing system. i and j are used to map to the i^{th} (respectively j^{th}) character in the source (respectively target) set of characters.

We can represent the relationship between the source and target language characters by making t_j 'depend' on s_i as shown in Figure 3.1. To complete the definition of the Bayesian network for this example, we need local probability models to represent the nature of the dependence of t_j on s_i. According to Figure 3.1, we need a probability model $P(s_i)$ to represent the distribution of the different characters in the source writing system. The distribution over the target random variable t_j is a conditional distribution which we denote here as $P(t_j|s_i)$. $P(t_j|s_i)$ means that for each assignment of values for the source random variable s_i, there is a different

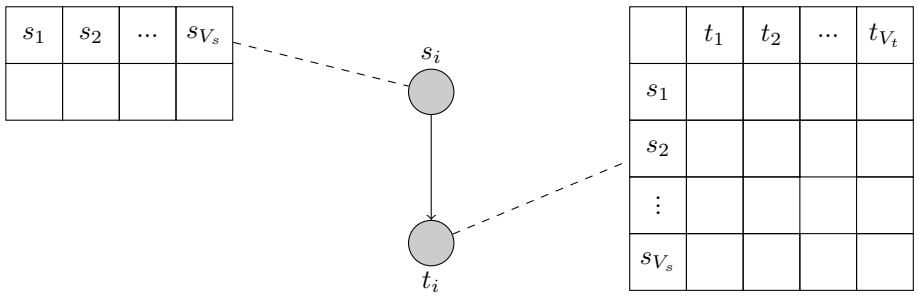

Figure 3.1: *A Bayesian network graph for relating characters between writing systems. s_i and t_j are random variables representing unique characters in the source (resp. target) writing system. V_s and V_t equal the total number of characters in the source (resp. target) writing system.*

distribution for t_j. For example, given that the variable s_i is assigned the English character 'a', there is a probability distribution for each Russian character t_j that appears as a corresponding character in a transliteration where 'a' appears.

3.2.3 Bayesian networks – inference

The main importance of a Bayesian network model is its use for answering queries related to the problem domain that it models. The reasoning process that is followed in using a Bayesian network model for answering a given query is refered to as *inference*. Generally, the inference task is to compute the posterior distribution over a subset of variables (*query variables*) given the values of some evidence variables. The computation can involve *hidden variables* which are neither *query* nor *evidence variables*. In the example of Figure 3.1 where the network models the similarity between characters in different writing systems, a query we may want to answer is what probability should be associated with assigning the variable t_j, some target character, given that s_i is assigned to some source character, for example $P(t_i = \text{ш}|s_i = \text{a})$ (in the case of a comparison of Latin 'a' and Cyrillic 'ш' characters). The value for this probability based on the network of Figure 3.1 is easy to obtain as we only need to look it up from the conditional probability distribution for the variable t_i. If the question is inverted, that is $P(s_i = \text{a}|t_i = \text{ш})$, we can still easily arrive to an answer by doing inference by enumeration using Bayes' rule:

$$P(s_i = \text{a}|t_i = \text{ш}) = \frac{P(s_i = \text{a}, t_i = \text{ш})}{P(t = \text{ш})} = \frac{P(s_i = \text{a}) \times P(t_i = \text{ш}|s_i = \text{a})}{\sum_{s_i} P(t_i = \text{ш}, s_i)} \tag{3.1}$$

Bayesian network inference done in this way leads to an exact answer and is therefore called *exact inference*. *Exact inference* requires a summation over a joint distribution in which we marginalize out irrelevant variables. The direct computation of probabilities as illustrated in Equation 3.1 is only possible for a small network. Several exact inference algorithms have been proposed to help increase efficiency over the direct approach while still being applicable for some complex representations. The following are some of the exact inference algorithms for Bayesian networks (Darwiche 2008): inference by *variable elimination*; inference by *tree clustering*; inference by *conditioning*; inference by *reduction to logic*.

As the size of the network increases with respect to the number of random variables and connections between them, we may experience an exponential blow up of the joint probability distribution represented by the model. All the exact inference methods mentioned above are sensitive to this complexity. Approximate inference algorithms are insensitive to this complexity and can be quite efficient regardless of the network topology.

3.2.4 Bayesian Networks - Limitation

The Bayesian network in Figure 3.1 can only be used from a static point of view. That is, the joint probability distribution over the variables s_i and t_i of the network is fixed although there are different values for $P(t_i|s_i)$. In a temporal setting such as the transliteration similarity task where we want to represent a distribution over a sequence of characters, the Bayesian network of Figure 3.1 cannot be used. In order to deal with problems where there is need to define distributions over more complex inter-relationships such as those in a temporal setting, template-based approaches have been proposed (Koller and Friedman 2009). In this thesis we use *Dynamic Bayesian Networks* as a template-based approach for modeling transliteration similarity.

3.3 Dynamic Bayesian Networks

Dynamic Bayesian networks have been developed to extend Bayesian networks to enable the representation and analysis of systems that change over time. The approach builds upon the framework of Bayesian networks but where random variables in the Bayesian network relate to time. DBNs are also usually discussed under three main aspects in the literature (Koller and Friedman 2009, Murphy 2002): representation, inference, and learning. We follow the same outline while using most of the notation as in Murphy (2002).

3.3.1 DBNs – representation

The possibility to have random variables relate to time in a Bayesian network enables DBNs to represent probability distributions over a sequence of random variables comprising of observations that are related to an underlying sequence of hidden states. A DBN model is formally defined as a pair $\langle \mathcal{B}_0, \mathcal{B}_{\rightarrow} \rangle$ where $\mathcal{B}_0$ is a Bayesian network over an initial distribution over states $P(Z_1^{(1:N)})$, and $\mathcal{B}_{\rightarrow}$ is a two-slice Temporal Bayes net (2-TBN) (Murphy 2002). Just as in Bayesian networks, the structure of a DBN is a directed acyclic graph (DAG) where each node represents a domain variable of interest at some time instant, and each directed arc represents the dependency between the two nodes it connects. A hidden state is represented in terms of a set of N_h random variables, S_t and the observation is also represented in terms of a set of N_o random variables, O_t. The transition and observation models of a DBN are defined as a product of the conditional probability distributions (CPDs) in the 2-TBN (Murphy 2002):

$$P(Z_t|Z_{t-1}) = \prod_{i=1}^{N} P(Z_t^{(i)}|\text{Pa}(Z_t^{(i)})) \tag{3.2}$$

where $Z_t^{(i)}$ is the i^{th} node at time t (which may be hidden or observed; and $N = N_h + N_o$), and $Pa(Z_t^{(i)})$ are the parents of $Z_t^{(i)}$, which may be in the same or previous time-slice (assuming a first-order Markov model). A given time t is associated with a number of states each of which is associated in turn with a number of parents $\text{Pa}(Z_t^{(i)})$ which may influence $Z_t^{(i)}$. The product of these characterizes the state at time t, Z_t.

For a given DBN, the general assumption is that parameters associated with nodes and dependencies among nodes are time invariant; and in particular, that the dependency parameters between two nodes across two time slices remain unchanged with time. Hence, as Murphy (2002) puts it, we can define a DBN over an observation sequence of length T by "unrolling" the 2-TBN until we have T time-slices. The joint distribution for the sequence of length T can then be obtained by multiplying together all of the CPDs (Murphy 2002):

$$P(Z_{1:T}^{(1:N)}) = \prod_{i=1}^{N} P_{\mathcal{B}_0}(Z_1^{(i)}|\text{Pa}(Z_t^{(i)})) \times \prod_{t=2}^{T}\prod_{i=1}^{N} P_{\mathcal{B}_{\rightarrow}}(Z_t^{(i)}|\text{Pa}(Z_t^{(i)})) \tag{3.3}$$

Before we can make the computation in equation 3.3, we need to have a DBN model and the stages required to construct one are similar to those described for Bayesian networks in section 3.2.1 on Bayesian network representation. That is, we need to specify suitable variables (Z) for the problem domain, the DBN structure

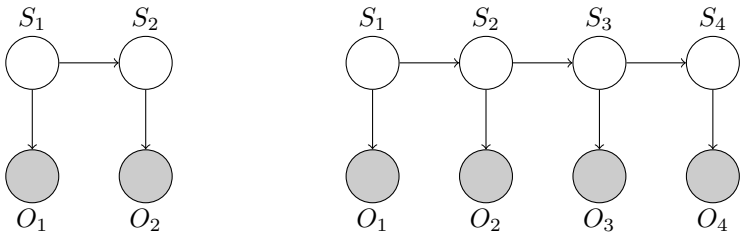

Figure 3.2: *(a) A 2-TBN for a Hidden Markov Model and (b) An unrolled network of four time-slices for the HMM. We follow the common convention of representing observed variables as shaded nodes and hidden variables as clear nodes.*

that encodes the relationship between the variables ($Z_i|\text{Pa}(Z_i)$), and the DBN model probabilities. A template is used to define a DBN structure and from the definition of a DBN and equation 3.3, the template may define at least two Bayesian networks: for $\mathcal{B}_0$ and for the 2-TBN $\mathcal{B}_{\rightarrow}$. Figure 3.2 is a graphical representation of a 2-TBN for a classic Hidden Markov Model (HMM) (a) and when unrolled for four time slices (b).

HMMs are categorized as the simplest of DBN models because they represent the hidden state using only one random variable. Many variants that extend the classic HMMs have also been proposed and used in various tasks. Murphy (2002) describes several examples as this long list shows: HMMs with mixture-of-Gaussians output, HMMs with semi-tied mixtures, auto-regressive HMMs, buried Markov models, mixed-memory Markov models, input-output HMMs, factorial HMMs, coupled HMMs, hierarchical HMMs, asynchronous IO-HMMs, variable-duration (semi-Markov) HMMs, mixtures of HMMs, segment models, and abstract HMMs. To this list, we add Pair HMMs which we propose to use for computing transliteration similarity. Although Pair HMMs are one of the two DBN approaches that we evaluate for transliteration detection and generation, some of the HMM variants in Murphy's (2002) list above actually warrant an empirical investigation to determine whether they can improve transliteration detection and generation quality over existing methods.

3.3.2 Transliteration example

Consider the name джейн written in Russian for which we would like to find a corresponding English name. Assuming that we use a Hidden Markov model to represent our transliteration problem. In Figure 3.2, we can think of each hidden state

as an English character that we would like to find given the model and the observed name in Russian. We can therefore think of an underlying event of English characters that should explain the observation of characters in a name written using the Cyrillic alphabet. We can further assume that in the process of finding a corresponding English representation, each English character maps to only one Russian character. The event of English characters may then proceed as follows. At the start, we can choose an English character according to an initial probability distribution given the Russian character д. Next, we choose a transition according to the distribution defined by the 2-TBN. Here, we would like to choose an English character that is the most likely explanation for the second Russian character given the English character in the previous step. We then proceed with the 2-TBN until all the Russian characters are explained. For this example, we need the starting probability distribution of English characters for each Russian character. We also need the distribution of moving from one English character in a previous step to an English character given a Russian character in the current step.

3.3.3 DBNs – Inference

In DBNs, the problem of inference is generally represented as the problem of finding the probability of hidden variables in a time-slice given a set of consecutive observations. The most important literature on DBN inference (Murphy 2002, Koller and Friedman 2009, Mihajlovic and Petkovic 2001) identifies four common DBN inference tasks: filtering, prediction, smoothing, and decoding. Let S_t and O_t respectively denote the state and observation at time t:

In filtering, at time t, inference is done to keep track of $P(S_t|O_{(1:t)})$, that is, we estimate the 'belief' state given all of the observations (evidence) obtained so far. Filtering is usually applied in estimating the state of a real time system given a set of values over an interval of time from some measurement function. The class of DBN models that have commonly been used for this purpose are Kalman Filter models. Filtering can also be useful in the context of transliteration as a sequence analysis problem. Given source and target language words as observation sequences and the parameters of a DBN model, filtering can be used to compute the probability distribution over the hidden states at the end of the sequence.

In prediction, given observations $O_{(1:t)}$, inference is done to predict the distribution over some subset of variables at time $t' > t$. Prediction can also be useful in the context of transliteration. For example, we can compute character sequence predictions by using DBNs to represent n-gram models.

Smoothing involves estimating a state of the past, given all the evidence up to the current time in some longer trajectory: $P(S_{t-l}|O_{1:t})$, where $l : \ 0 \leq l \leq t$ can vary

with different time ranges. Smoothing in this case is aimed at incorporating future evidence to help reduce temporary fluctuations in the belief state which can lead to temporary "misconceptions" in the belief state (Koller and Friedman 2009).

In decoding, the aim is to find the most likely sequence of hidden states given the observations: $S'_{1:t} = \underset{S_{1:t}}{\text{argmax}} P(S_{1:t}|O_{1:t})$. This inference task is important for both transliteration detection and generation. In transliteration detection, we can use the probability score associated with the most likely alignment sequence between a source word and a candidate transliteration for comparison with other candidate transliterations. In transliteration generation, we can assume target words to be associated with observations and decoding can then be used to infer the sequence of source language characters that explain the observed target word characters.

One additional task that is important for transliteration detection is the computation of the probability for an observed pair of source and target language words which can be used for evaluation with other candidate words. In the previous paragraph we have mentioned one way of achieving that - by comparing the scores associated with the 'best' alignments of a source word and candidate target transliterations. The other way involves a summation over all possible hidden trajectories. For the classic HMMs, this can be achieved by using a forward algorithm (Rabiner 1989).

3.3.4 DBNs – Learning

There are two types of learning that are associated with DBNs: DBN structure learning and DBN parameter estimation. In DBN structure learning, the task is to extract a DBN structure as well as its parameters given training data. In DBN parameter estimation, we assume that the DBN structure is known and the learning task is to determine the parameters that define the conditional probability distributions of the attributes.

a) DBNs – structure learning

Since a DBN can be represented by two networks $\mathcal{B}_0$ and $\mathcal{B}_{\rightarrow}$, learning the structure of a DBN reduces to learning the structure of $\mathcal{B}_0$ and $\mathcal{B}_{\rightarrow}$. Boyen et al. (1999) provide a detailed introduction to DBN structure learning. Here, we briefly point out the most relevant parts of their discussion.

If we have complete data (that is the training sequence $\mathcal{D}$ is fully observable), the learning task is to find the networks $\mathcal{B}_0$ and $\mathcal{B}_{\rightarrow}$ that "best match" $\mathcal{D}$ (Friedman et al. 1998). The notion of best match is defined using a scoring function and the term of interest is the *log-likelihood function*, defined as $\mathcal{L}(\mathcal{B}_0, \mathcal{B}_{\rightarrow} : \mathcal{D}) = \log P(\mathcal{D}|\mathcal{B}_0, \mathcal{B}_{\rightarrow})$. The *log-likelihood* function measures how likely the data is given the candidate models

$\mathcal{B}_0, \mathcal{B}_{\rightarrow}$. The *log-likelihood* function relies on *sufficient statistics* that summarize the frequencies of the relevant events in the data. A scoring function that utilizes the *log-likelihood* function can be defined and the goal then is to find the networks that maximize the score.

If we have incomplete data (that is, the training sequence $\mathcal{D}$ is partially observable), then we no longer know the exact counts in the data. As one of the earliest attempts at DBN structure learning, Friedman (1997) extends the traditional *Expectation Maximization* (EM) algorithm to a *Structural EM* (SEM) algorithm. The Expectation step (E-step) in the SEM algorithm is similar to that in the traditional EM algorithm. The E-step uses the current structure and parameters to complete the data and compute expected counts (*expected sufficient statistics*). The Maximization step (M-step) of the SEM algorithm re-estimates parameters and also evaluates candidate structures using *expected sufficient statistics* computed from the current structure.

Although it is important to undertake an empirical investigation into DBN structure learning for transliteration detection and generation, it will not be part of the work reported in this thesis. Instead, we will evaluate known DBN structures that we have adapted from related work and those that we have modified and specified for computing transliteration similarity. Therefore, in learning DBN models for computing transliteration similarity in this thesis, our main concern is with DBN parameter estimation.

b) DBNs – parameter learning

In DBN parameter estimation, we start with some apriori knowledge about the DBN model which is represented in the form of a prior probability distribution over model parameters (Ghahramani 1998). The knowledge is updated using data to obtain a posterior probability distribution over models and parameters. Assuming that the prior probability distribution over the parameters for a given DBN model structure is specified by $P(\theta|\mathcal{B})$, a data set $\mathcal{D}$ is used to compute a posterior distribution over the parameters as shown in Equation 3.4 below.

$$P(\theta|\mathcal{B}, \mathcal{D}) = \frac{P(\mathcal{D}|\theta, \mathcal{B})P(\theta|\mathcal{B})}{P(\mathcal{D}|\mathcal{B})}. \tag{3.4}$$

The approach used in estimating DBN model parameters is also determined by one of the two cases described in the previous section on DBNs structure learning. That is whether learning is based on complete data or incomplete data. If the training sequence is $\mathcal{D}$ is fully observable and we assume a DBN structure, the goal of learning is to estimate DBN model parameters (θ) that 'best match' $\mathcal{D}$. Here, we also use a *log-likelihood* function to measure the likelihood of the $\mathcal{D}$ given the parameters (θ)

and a scoring function that facilitates the estimation of parameters that maximize the likelihood value.

If the training sequence is partially observable, an Expectation Maximization algorithm is often used. The DBN models we propose to use for transliteration detection and generation involve hidden states or variables for which we apply an EM algorithm to estimate DBN model parameters. Although the EM algorithm is well covered in the Literature, we provide a detailed review in the following subsection since it plays the most important role in the training of the DBN models that we propose to use for computing transliteration similarity in transliteration detection and generation.

c) DBN Parameter learning using the EM algorithm and its generalization

Maximum Likelihood Estimation (MLE) using an Expectation Maximization (EM) algorithm is the most common aproach for estimating parameters given that a particular model has hidden or latent variables. Our review of the EM algorithm is completely based on an unpublished note by Stuart Russell and on a tutorial by Borman (2004) which also builds upon Stuart Russell's note. We also use similar notation as in Russell's note and in Borman's (2004) tutorial.

The EM algorithm iterates between two steps: the Expectation step where we compute values of the model's hidden and observed variables given training data and current model parameters; and a Maximization step (M-step), where new model parameters are estimated that maximize the likelihood of training data. The overall goal when using an EM algorithm is to find the model parameters θ such that $P(\mathcal{D}|\theta)$ is maximal. The M-step of the EM algorithm can be achieved by introducing a *log likelihood function* of the parameters θ ($\mathcal{L}(\theta)$) given the data $\mathcal{D}$ as follows:

$$\mathcal{L}(\theta) = \ln P(\mathcal{D}|\theta) \tag{3.5}$$

Assuming that θ_n denotes the current estimate for the model's parameters after n iterations, we wish to find an estimate of the parameters in the next iteration (θ_{n+1}) such that the difference of the values of the likelihood functions for the current and next iteration $\mathcal{L}(\theta_{n+1}) - \mathcal{L}(\theta_n)$ is maximized. In the presence of a set of hidden variables in the model, denoted by Z, We can re-write $P(\mathcal{D}|\theta_{n+1})$ as follows:

$$P(\mathcal{D}|\theta_{n+1}) = \sum_z P(\mathcal{D}|z, \theta_{n+1}) P(z|\theta_{n+1}) \tag{3.6}$$

where z refers to the values of Z. The difference that we wish to maximize can then be writen as:

$$\mathcal{L}(\theta_{n+1}) - \mathcal{L}(\theta_n) = \ln \left(\sum_z P(\mathcal{D}|z, \theta_{n+1}) P(z|\theta_{n+1}) \right) - \ln P(\mathcal{D}|\theta_n). \tag{3.7}$$

As Equation 3.7 involves the logarithm of a sum, we can use Jensen's inequality (Jensen 1906). The inequality is stated as follows:

$$\ln \sum_{i=1}^{n} \lambda_i x_i \geq \sum_{i=1}^{n} \lambda_i \ln(x_i) \qquad \text{where } \lambda_i \geq 0 \text{ are constants such that } \sum_{i=1}^{n} \lambda_i = 1.$$

If we introduce constants of the form $P(z|\mathcal{D}, \theta_n)$ to Equation 3.7 such that $P(z|\mathcal{D}, \theta_n) \geq 0$ and $\sum_z P(z|\mathcal{D}, \theta_n) = 1$, we can apply Jensen's inequality as follows:

$$\begin{aligned}
\mathcal{L}(\theta_{n+1}) - \mathcal{L}(\theta_n) = \quad & \ln \left(\sum_z P(\mathcal{D}|z, \theta_{n+1}) P(z|\theta_{n+1}) \cdot \frac{P(z|\mathcal{D}, \theta_n)}{P(z|\mathcal{D}, \theta_n)} \right) - \ln P(\mathcal{D}|\theta_n) \\
= \quad & \ln \left(\sum_z P(z|\mathcal{D}, \theta_n) \frac{P(\mathcal{D}|z, \theta_{n+1}) P(z|\theta_{n+1})}{P(z|\mathcal{D}, \theta_n)} \right) - \ln P(\mathcal{D}|\theta_n) \\
\geq \quad & \sum_z P(z|\mathcal{D}, \theta_n) \ln \left(\frac{P(\mathcal{D}|z, \theta_{n+1}) P(z|\theta_{n+1})}{P(z|\mathcal{D}, \theta_n)} \right) - \ln P(\mathcal{D}|\theta_n) \\
= \quad & \sum_z P(z|\mathcal{D}, \theta_n) \ln \left(\frac{P(\mathcal{D}|z, \theta_{n+1}) P(z|\theta_{n+1})}{P(z|\mathcal{D}, \theta_n) P(\mathcal{D}|\theta_n)} \right) \\
\triangleq \quad & \triangle(\theta_{n+1}|\theta_n). \qquad (3.8)
\end{aligned}$$

Equation 3.8 can be written as

$$\mathcal{L}(\theta_{n+1}) \geq \mathcal{L}(\theta_n) + \triangle(\theta_{n+1}|\theta_n) \qquad (3.9)$$

Equation 3.9 shows that $\mathcal{L}(\theta_n) + \triangle(\theta_{n+1}|\theta_n)$ is bounded above by $\mathcal{L}(\theta_{n+1})$. It is easy to show that any θ_{n+1} which increases $\mathcal{L}(\theta_n) + \triangle(\theta_{n+1}|\theta_n)$ will also increase $\mathcal{L}(\theta_{n+1})$. In order to maximize $\mathcal{L}(\theta_{n+1})$, the EM algorithm requires the selection of θ_{n+1} such that $\mathcal{L}(\theta_n) + \triangle(\theta_{n+1}|\theta_n)$ is maximized. If we denote the maximization value by θ'_{n+1}, then

$$\begin{aligned}
\theta'_{n+1} &= \underset{\theta_{n+1}}{\text{argmax}} \{\mathcal{L}(\theta_n) + \triangle(\theta_{n+1}|\theta_n)\} \\
&= \underset{\theta_{n+1}}{\text{argmax}} \left\{ \mathcal{L}(\theta_n) + \sum_z P(z|\mathcal{D}, \theta_n) \ln \frac{P(\mathcal{D}|z, \theta_{n+1}) P(z|\theta_{n+1})}{P(\mathcal{D}|\theta_n) P(z|\mathcal{D}, \theta_n)} \right\} \\
&= \underset{\theta_{n+1}}{\text{argmax}} \left\{ \sum_z P(z|\mathcal{D}, \theta_n) \ln P(\mathcal{D}|z, \theta_{n+1}) P(z|\theta_{n+1}) \right\}
\end{aligned}$$

$$= \underset{\theta_{n+1}}{\operatorname{argmax}} \left\{ \sum_{z} P(z|\mathcal{D}, \theta_n) \ln \frac{P(\mathcal{D}, z, \theta_{n+1})}{P(z, \theta_{n+1})} \frac{P(z, \theta_{n+1})}{P(\theta_{n+1})} \right\}$$

$$= \underset{\theta_{n+1}}{\operatorname{argmax}} \left\{ \sum_{z} P(z|\mathcal{D}, \theta_n) \ln P(\mathcal{D}, z|\theta_{n+1}) \right\}$$

$$= \underset{\theta_{n+1}}{\operatorname{argmax}} \{ E_{Z|\mathcal{D},\theta_n} \{ \ln P(\mathcal{D}, z|\theta_{n+1}) \} \} \tag{3.10}$$

Equation 3.10 defines both the expectation and maximization steps. In the expectation step, the algorithm determines the conditional expectation

$$E_{Z|\mathcal{D},\theta_n} \{ \ln P(\mathcal{D}, z|\theta_{n+1}) \};$$

and in the maximization step, the algorithm maximizes the expression using θ_{n+1}.

The Generalized EM algorithm relaxes the requirement of maximizing $\triangle(\theta_{n+1}|\theta_n)$ to the one of increasing $\triangle(\theta_{n+1}|\theta_n)$ so that $\triangle(\theta'_{n+1}|\theta_n) \geq \triangle(\theta_n|\theta_n)$. With this requirement it is possible to show that the likelihood $\mathcal{L}(\theta_{n+1})$ is guaranteed to be non-decreasing at each iteration.

3.4 Conclusion

The framework of Dynamic Bayesian Networks (DBNs) offers a large space of models which can be exploited to represent and reason about various temporal domains. However, the use of DBNs and Bayesian networks in general is more likely to be successful if the set of variables and the interactions between variables that are defined in the first stages of model construction do provide an adequate representation of the domain being modeled. Our proposal to use DBNs in transliteration related tasks can proceed in different ways ranging from determining DBN model structures given example transliteration data to using already specified DBN structures. In the thesis we are concerned with the latter where we investigate DBN structures that have already been proposed but not yet tested in transliteration-related tasks. Given a DBN model structure, the parameters associated with the model also need to be estimated before we can use the model to answer queries associated with a given domain. In this chapter, we have reviewed the expectation maximization algorithm which has been and still is the cornerstone of parameter estimation in the framework of DBNs. The estimation of DBN model parameters forms the first main phase in our application of DBN models in transliteration-related tasks. Given a fully parameterized DBN model, we can use a suitable inference algorithm to answer a specific type of query in the problem domain. In transliteration mining, we would like to use

the model to compute the similarity estimate associated with a source and target word. Specifically, we want to use the model to compute the probability of observing the pair as candidate and transliteration. In transliteration generation, we would like to use the DBN model parameters for suggesting hypothetical target representations given a source word. The use of a particular inference algorithm to answer a specific query with regard to transliteration mining and generation constitutes the other main phase in applying the DBN models.

Chapter 4

Pair HMMs for transliteration detection

4.1 Introduction

The detection of transliterations requires an analysis on words written in the source and target languages with the aim of determining word(s) from the target language that are the most likely representation(s) of a word in a source language and vice versa. In this thesis, we assume the transliteration detection process to be composed of two steps. The first step involves computing the transliteration similarity between a source language word and a target language candidate transliteration or between a candidate original source word a known transliteration in a target language. The second step involves making a decision on whether or not to regard the pair of source and target language words as a true transliteration pair based on their computed transliteration similarity.

In this chapter, we introduce the approach of Pair Hidden Markov models (Pair HMMs) as the first of two Dynamic Bayesian Network (DBN)-related approaches that we have proposed to use for computing transliteration similarity in the process of detecting transliterations in bilingual text. Pair HMMs as the name suggests extend the classic Hidden Markov models (HMMs) by modeling two observation sequences instead of one sequence. The inference algorithms for the Pair HMMs are also modifications of the traditional inference algorithms (that is Forward-Backward, Viterbi) for the classic HMMs. The Pair HMM approach in its own right offers a huge model space, but we choose to start our investigation with some structures and

This chapter is an extended version of the following publications:
P. Nabende, J. Tiedemann, and John Nerbonne – Pair Hidden Markov Model for Named Entity Matching, *Innovations and advances in Computer Sciences and Engineering*, pp. 497–502, 2010, Springer Netherlands; and
P. Nabende – Comparison of applying Pair HMMs and DBN models in transliteration identification, *Proceedings of the* 20^{th} *Computational Linguistics in Netherlands meeting*, pp. 107–122, Feb 2010, Utrecht, The Netherlands.

parameter definitions that have been successfully used for computing word similarity in tasks having requirements similar to those for computing transliteration similarity such as cognate identification (Mackay and Kondrak 2005) and dialect comparison (Wieling et al. 2007). Our aim here is to determine whether assumptions that were used in Pair HMMs for computing word similarity in previous work are valuable for computing transliteration similarity. We then propose additional Pair HMM parameter settings with the aim of determining the effect of parameter changes on transliteration detection accuracy. For preliminary experiments we used our own-prepared transliteration data consisting of geographic name pairs extracted from the Geonames online database for four language pairs: English-Dutch, English-French, English-German, and English-Russian. In the second set of transliteration detection experiments, we used standard transliteration data for seven language pairs from the 2009 and 2010 Named Entities Workshop (NEWS) shared tasks on transliteration generation for evaluating several Pair HMMs and a standard baseline approach that uses pair n-gram information for computing transliteration similarity. The seven language pairs include: English-Bengali, English-Chinese, English-Hindi, English-Kannada, English-Russian, English-Tamil, and English-Thai.

In the following section, we introduce the classic HMMs and briefly review recent uses of the HMM framework in modeling transliteration. From there, we introduce Pair HMMs starting with their origins to the requirements for applying them to compute transliteration similarity. We then describe the Pair HMMs that we use to investigate the effects of parameter changes on transliteration detection quality. Based on atleast one of the Pair HMM structures that we have proposed to investigate, we also describe the algorithms for training the Pair HMMs and for computing transliteration similarity.

4.2 Hidden Markov Models

4.2.1 A brief review on representation

Hidden Markov models find their origins as extensions to Markov models. Rabiner (1989) describes how the concept of Markov models can be extended to include the case where the observation is a probabilistic function of a state, resulting in a model that is a doubly embedded stochastic process in which the underlying stochastic process is not observable (i.e. it is *hidden*). Coupled with the defining property that the underlying stochastic process satisfies the *Markov property*, we end up with a *Hidden Markov Model*. To satisfy the *Markov property*, the value of the state at time t (denoted here by S_t), is dependent on only the previous state (S_{t-1}), and independent of all other states prior to $t-1$. The outputs from the states also satisfy

the Markov property. That is, the observation in a state at time t (denoted here by O_t) is independent of all other states and observations. Taken together, these Markov properties lead to the following factorization of the joint distribution of a sequence of states (S) and observations (O) (Ghahramani 2001):

$$P(S_1, ..., S_T, O_1, ..., O_T) = P(S_1)P(O_1|S_1)\prod_{t=2}^{T} P(S_t|S_{t-1})P(O_t|S_t) \tag{4.1}$$

This factorization of the joint distribution can be represented graphically as a Dynamic Bayesian network (see Figure 3.2 in Chapter 3). It is clear from equation 4.1 that to determine the probability distribution over sequences of observations, we need probability distributions over: the initial state $P(S_1)$, the $K \times K$ transition matrix defining $P(S_t|S_{t-1})$, and the output (or emission) model defining $P(O_t|S_t)$.

4.2.2 Recent use of HMMs in machine transliteration

The transition and output model $P(O_t|S_t)$ in equation 4.1 can be specified in various ways. In the following, we review some recent formulations for the HMM transition and output models in the context of the transliteration modeling process.

a) Bi-Stream HMMs

Zhao et al. (2007) propose a bi-stream HMM for letter-alignment within named entity (NE) pairs. When using the bi-stream HMM, the probability of a source NE (denoted by s_1^I) given a target NE (denoted by t_1^J), is formulated in equation 4.2 below as:

$$P(s_1^I|t_1^J) = \sum_{a_1^I} \prod_{i=1}^{I} P(s_i|t_{a_i})P(c_{s_i}|c_{t_{a_i}})P(a_i|a_{i-1}) \tag{4.2}$$

where a_i maps s_i to the target letter t_{a_i} at position a_i in the target NE. $P(a_i|a_{i-1})$ is the transition probability distribution; $P(s_i|t_{a_i})$ is a letter-to-letter translation lexicon; c_{s_i} is a letter cluster of s_i and $P(c_{s_i}|c_{t_{a_i}})$ is a cluster level translation lexicon. The bi-stream HMM generates two streams of observations: the letters together with their classes following the distribution for $P(s_i|t_{a_i})$ and $P(c_{s_i}|c_{t_{a_i}})$ at each state respectively. Zhao et al. (2007) also define a constraint to ensure that the transition can only jump forward or stay at the same state.

b) Maximum n-gram HMMs

Zhou (2009) views an HMM as a bi-gram model where the transliteration of the current character is dependent on the transliteration of a previous character. In this

approach, the underlying hidden process is a sequence of characters in one language which generates characters in the other language. When using the bi-gram HMM, the probability of target NE given the source NE is formulated in equation 4.3 below as:

$$P(t_1^n|s_1^n) = P(t_1|s_1)P(t_2|s_2,t_1)...P(t_n|s_n,t_{n-1}) \tag{4.3}$$

where $$P(t_i|s_i) = \frac{\#\text{ of times } s_i \text{ translates to } t_i}{\#\text{ of times } s_i \text{ occurs}} \quad \text{and}$$

$$P(t_i|s_i,t_{i-1}) = \frac{\#\text{ of times } s_i \text{ translates to } t_i \text{ given } s_{i-1} \rightarrow t_{i-1}}{\#\text{ of times } s_{i-1} \text{ translates to } t_{i-1}}$$

Zhou uses an alignment procedure to build a translation lexicon that is in turn used for obtaining character translation pair occurrences. This facilitates the computation of the probabilities in equation 4.3. It is straightforward to formulate the equations for estimating transliteration probability for the trigram HMM case and other higher order n-gram HMMs.

c) HMMs for searching transliterations

Darwish (2010) uses an approach similar to the bigram HMM approach above. In Darwish's case, a character sequence (denoted here by s'_i in the source NE (s_1^I) is taken to be a potential transliteration for a character sequence t'_j in the target NE (t_1^J). Darwish calculates the probability of s'_i given t'_j from a trained model as follows:

$$P(t'_j|s'_i) = \prod_{\text{all } s_x...s_y} P(s_x...s_y|t'_k...t'_l) \tag{4.4}$$

where $s_x...s_y$ are non-overlapping segments generated by finding all 2^{n-1} segmentations of the character sequence s'_i. According to Darwish (2010), the segmentation producing the highest probability is chosen, and all target segment sequences $t'_k,...,t'_l$ that are known to produce $s_x...s_y$ for each possible segmentation are also produced. If a set of non overlapping sequences of $t'_k,...,t'_l$ generates t'_j, then t'_j is identified as a transliteration of s'_i; and if multiple target sequences have $P(s'_i|t'_j) > 0$, then the t'_j that maximizes $P(s'_i|t'_j)$ is chosen as the proper transliteration.

As the review in chapter 3 on Dynamic Bayesian Networks showed, there are several other types of HMMs that have been proposed and used successfully in various tasks and yet they might also be useful for transliteration detection. In this chapter, we only investigate the edit distance-based approach of Pair HMMs which is also different from the HMM approaches reviewed above. Our motivation in investigating the Pair HMM approach is based on our observation of its success in various tasks ranging from biological sequence analysis through to the Natural Language Processing (NLP) task of computing word similarity which is similar to that of computing transliteration simiarity.

4.3 Pair Hidden Markov Models

4.3.1 Origins

The approach of Pair HMMs originates from modifications to a pairwise alignment finite state automaton (Durbin et al. 1998). The conversion of the automaton to a Pair HMM is achieved by defining a model which fulfills the representational requirements of HMMs and whose parameters are defined in such a way so as to approximate the parametric definitions of the finite state automaton. Durbin et al. (1998) define three emission states that correspond to the states of the automaton as follows: a match state (denoted by M) which has emission probability distribution p_{ab} for emitting an aligned pair of symbols a:b; and two gap states (denoted by X and Y) with distributions q_a for emitting a symbol a against a gap. Durbin et al. (1998) also satisfy the requirement that the probabilities for all the transitions leaving each state sum to one. Figure 4.1 illustrates Durbin et al.'s (1998) initial probabilistic version of the pairwise alignment finite state automaton. This probabilistic model is similar to a Hidden Markov model but instead of emitting a single sequence, it emits a pair of sequences.

The model in Figure 4.1 allows for symmetry between source and target and therefore uses two parameters to represent transition probabilities between emission states. These parameters are denoted by δ (which represents the transition probability from the *match* state to a *gap* state) and ϵ (which represents the transition probability of staying in a *gap* state). Durbin et al. also define the start and end states to formalize conditions for initialization and termination. They specifically define transition probabilities from the *start* state to be the same as transition probabilities from the *substitution* state to any of the *emitting* states. They also define the probability of transition to the end state from each *emitting* state (τ) be the same. Durbin et al. (1998) then define the inference and learning algorithms for the proposed Pair HMM structure in the context of biological sequence analysis. The reader is referred to Durbin et al. (1998) for a detailed description of the accompanying Pair HMM algorithms based on the model structure in Figure 4.1.

4.3.2 Pair HMMs for modeling word similarity

After the introduction of the Pair HMM approach, Mackay and Kondrak (2005) proposed to adapt it to compute word similarity and use it in a cognate identification task. Mackay and Kondrak proposed a number of modifications to Durbin et al.'s (1998) original Pair HMM structure so as to suit the alignment and comparison of words in natural language. In the first modification, they added a pair of transitions between the gap states X and Y each having the same transition probability (denoted

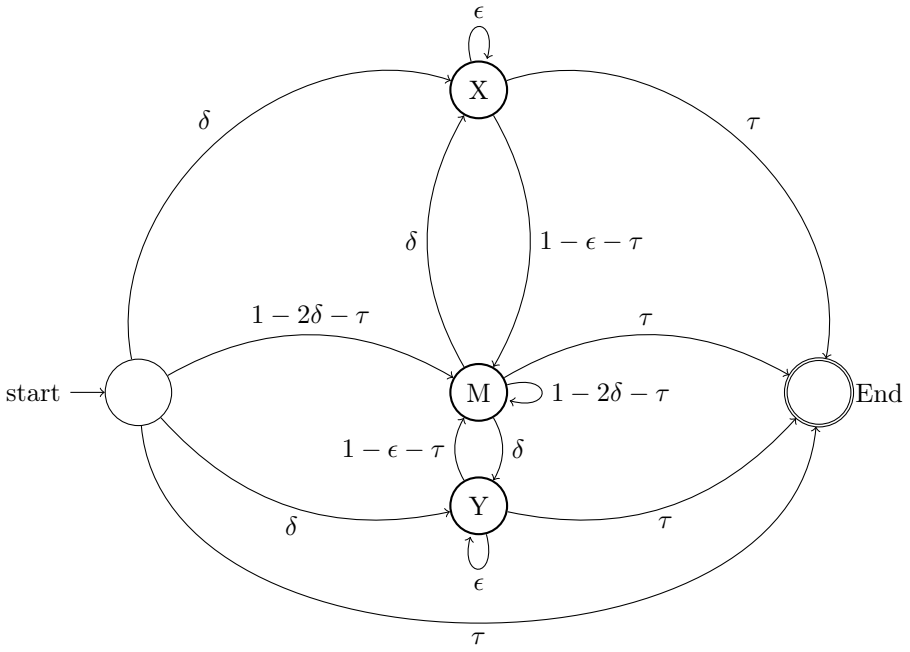

Figure 4.1: *Probabilistic version of the pairwise alignment finite state automaton in Durbin et al. (1998). Three transition parameters are specified including δ (for transitions from the match state (M) to gap states (X or Y)), ϵ (for transitions of staying in a gap state, and τ (for transitions to the end state).*

by λ). In another modification, they defined two parameters to represent two different transition probabilities to the end state. They defined τ_M to represent the transition probability from the *match* state to the end state, and τ_{XY} to represent the transition probability from the gap states to the end states. Figure 4.2 shows a finite state representation of the Pair HMM proposed by Mackay and Kondrak (2005). Mackay and Kondrak then modified the Pair HMM inference algorithms based on the Pair HMM structure in Figure 4.2. The reader is referred to Mackay's thesis (2004) for a further description of the inference and learning algorithms for that particular Pair HMM structure.

4.3.3 Pair HMMs for modeling transliteration similarity

The task of computing transliteration similarity is similar to that of computing word similarity where the approach of Pair HMMs has been applied successfully (Mackay

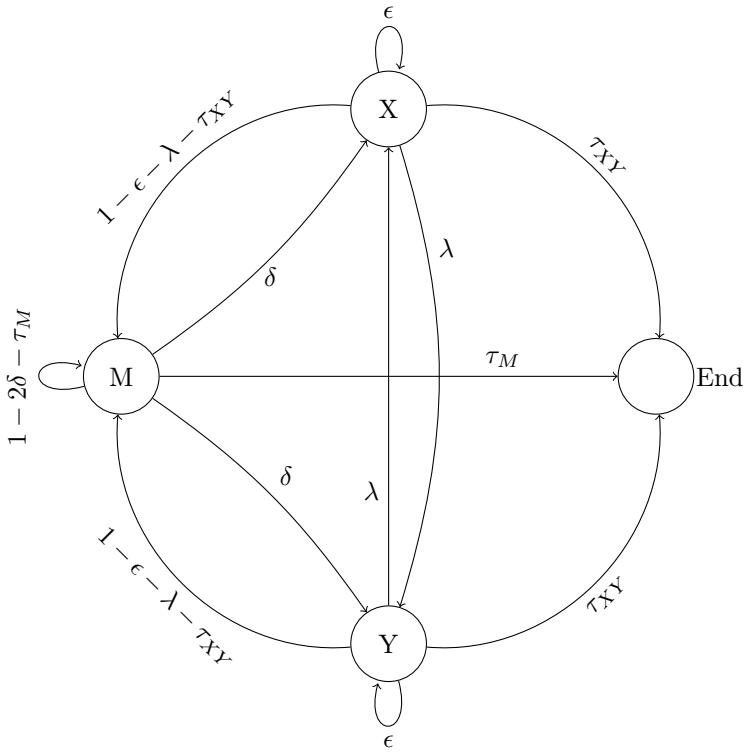

Figure 4.2: *Finite state representation of the Pair HMM proposed by Mackay and Kondrak (2005) for computing word similarity. The Pair HMM uses five transition parameters: three for the transition probabilities between edit states (δ, λ, ϵ), and two to the end state (τ_{XY}, τ_M).*

and Kondrak 2005, Wieling et al. 2007). However, we need to know whether or not the Pair HMMs approach is useful for computing transliteration in the process of detecting transliterations from bilingual text. Our starting point is to cast the task of computing transliteration similarity in exactly the same way as that of computing word similarity. Before we do that, we need to address requirements for computing transliteration similarity. Some of the requirements have already been addressed by Mackay (2004) in their use of Pair HMMs for computing word similarity.

The first requirement involves representing the source and target language words in a form that enables inference to be done efficiently using Pair HMMs. In transliteration, the source and target language words are transcribed using different writing systems. If we consider the orthographic representation of the words, the method

used for computing transliteration similarity should be able to handle the different alphabets. Alternatively, we might use a phonetic alphabet to transcribe the words phonetically. If we use a phonetic representation, the transliteration similarity estimation method can assume one alphabet for both source and target languages. However, the requirement to use a phonetic representation is very likely to be difficult due to lack phonetically transcribed data. Although transliteration is commonly defined to involve a phonetic transformation of a source language word to a target language word using a different writing system, some approaches that use orthographic-only representations are reported to have resulted in comparable if not better machine transliteration accuracy than phoneme-based approaches (Li et al. 2004).

The other requirement involves the tokenization of the source and target language words in their representational form into segments that can be used for alignment. A starting point for tokenization, is to use the language's alphabet (orthographic representation) or a phonetic alphabet (phonetic representation). It is also common to find a combination of characters in one language corresponding to a single character in the other language: an example is the English letter pair ⟨ch⟩ as compared to the Russian representation ⟨ч⟩. In that case it should be possible to represent ⟨ch⟩ as a single token on the English side.

There are some assumptions we can consider to simplify the application of the Pair HMM method to transliteration similarity estimation. We take a leaf from the word similarity estimation task in Mackay (2004) which uses a number of these assumptions for applying Pair HMMs in the cognate identification task. Since the transliteration detection task involves the analysis of NEs, we can regard an NE as a particular 'type of word'. We therefore hardly expect some of the assumptions used for computing word similarity to affect transliteration detection quality. First, we assume a monotonic ordering of characters in the source and target language words when computing transliteration similarity. That is the basic ordering of the tokens remains the same between the source and target language. Second, we assume no crossing links in the token alignments between the source and target words. Third, we assume only one-to-one character alignments. This third assumption critically limits the application of Pair HMMs in detecting transliterations for some language pairs whose writing systems are fundamentally different such as between English and Chinese. English uses a phonemic alphabet whereas the writing system for Chinese is mostly logographic[1]. Consider this example where we have the name "Peter" as written in English and one of its simplified Chinese representations 彼得. If we tokenize the names by character, and try to get an alignment for them, we will always end up with atmost two character matching alignments while the rest of the characters are aligned to an empty string. Consequently, the resulting Pair

[1] In a logographic writing system, each symbol in theory represents an idea.

HMM will have a poor representation with regard to strings in the source and target language. For this specific case, we would like to use a method that matches the Chinese character '彼' to the English character sequence 'Pe', and also matches '得' to 'ter' which is close to a true representation of the character correspondences in the source and target language strings.

Mackay and Kondrak (2005) also maintain some symmetries associated with the gap states "X" and "Y" of a Pair HMM. The same symmetries are also maintained in (Wieling et al. 2007) in the Dutch dialect comparison task. For the dialect comparison task, it is indeed more meaningful to consider the states "X" and "Y" the same since the source and target words are from the same language. For the transliteration task, where the source and target languages use different writing systems, the gap states should be distinct and should reflect the different properties of the source and target writing systems. In our preliminary experiments, we determine the effect on transliteration detection quality from using Pair HMMs that distinguish between the the gap states based on different alphabets for the source and target language in comparison to using Pair HMMs where the gap states are assumed to be the same. In the former, we constrain the the Pair HMMs to generate the pair of words based on the different probability distributions in the gap states X and Y that reflect the distinct properties between the source and target writing system respectively. In the latter, the Pair HMMs generate the pair of words based on a probability distribution for X which is the same for Y for which we assume that the writing system for the source and target languages is the same and can be obtained by combining any distinct writing systems.

When using Pair HMMs for a particular task, the main focus is usually on comparing the effectiveness of different Pair HMM inference and learning algorithms and determining the optimal structure of the underlying model. To determine the optimal structure, we can examine the relative contribution of three sets of parameters (Mackay and Kondrak 2005): substitution parameters, gap parameters (insertion and deletion), and transition parameters. Because substitution parameters constitute the core of a Pair HMM, focus is usually put on the gap and transition parameters. In the second set of transliteration detection experiments, we determine the effect of Pair HMM transition parameters on transliteration detection quality. In our investigation, we have defined three Pair HMMs in addition to Mackay and Kondrak's (2005) Pair HMM where we train the models to use different probability distributions for the gap states (X and Y) that reflect differences in the writing systems for the source and target languages. In the first case, we train a Pair HMM without transition parameters between each of the edit states; the only transition parameters in the Pair HMM are from a from a start state to one of the edit states and from one of the edit states to the end state. In the second case (Figure 4.3), we define the

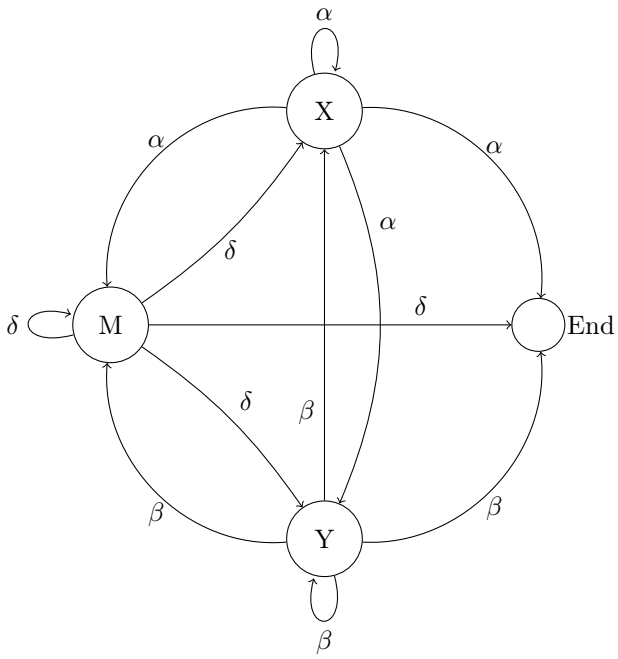

Figure 4.3: *A finite state representation of a Pair HMM with three transition parameters (α, β, δ). The probability of leaving an edit state to another state is the same for all destination states.*

Pair HMM to use three transition parameters associated with leaving one of the edit states, and where we also specify the probability of starting in one of the edit states to be the same as the probability of moving from the substitution state (M) to that edit state including M. In the third case (Figure 4.4), we define the Pair HMM to use different probabilities for transitions between the edit states and to the end state.

4.3.4 Pair HMMs – Inference

As is the case for the classic HMMs, we are concerned with three important tasks which are necessary in using Pair HMMs for computing transliteration similarity. estimation:

The first task, which is also related to our aim for using Pair HMMs, is to compute the probability of a pair of words given a specific Pair HMM. In the context of transliteration detection, this task is actually that of computing transliteration similarity. The probabilities that we compute from this task enable the comparison of candidate transliterations given an original source language word and vice versa

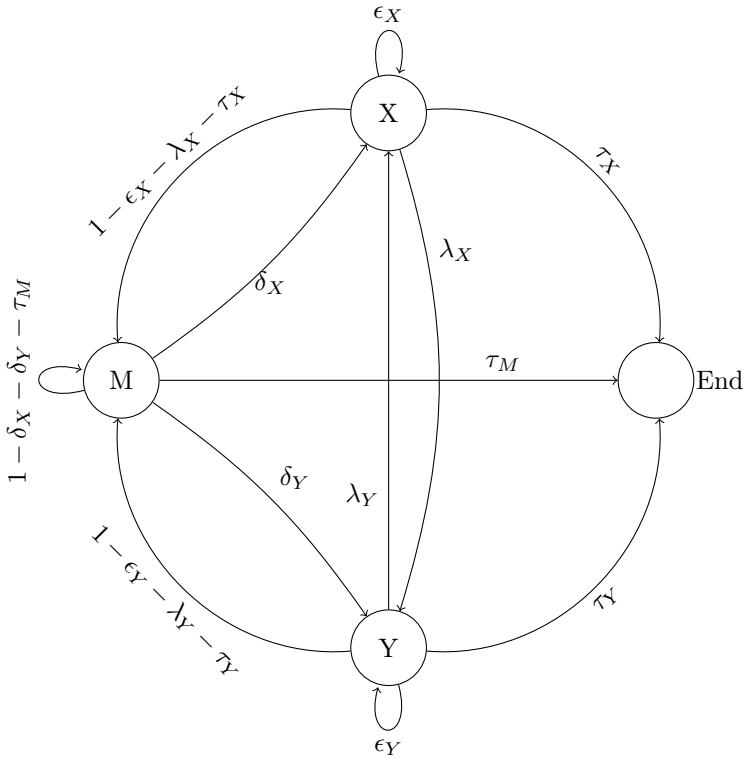

Figure 4.4: *A finite state representation of a Pair HMM with nine distinct transition parameters: six for transition probabilities between the edit states (δ_X, δ_Y, λ_X, λ_Y, ϵ_X, ϵ_Y), and three to the end state from each edit state (τ_X, τ_Y, and τ_M).*

and hence the decision on which pairs to regard as 'true transliteration pairs'. For this task, we evaluate the Forward and Viterbi algorithms and their log-odds versions which combine the base algorithms with a random Pair HMM (see below).

The second task is to determine the most probable alignment given a pair of words and a Pair HMM. In this task we need to use a version of the Viterbi algorithm for the given Pair HMM to find the most probable alignment sequence given two words. Given training data, we can use the alignment sequences to estimate parameters of a Pair HMM. Given a Pair HMM, we can use the alignment score to represent transliteration similarity.

The third task is associated with estimating the parameters of a Pair HMM. Given a Pair HMM structure with unspecified parameters, we need to compute Pair HMM parameters that maximize the likelihood of data which in our case consists of true

transliteration pairs. There are a number of algorithms that we can use to estimate Pair HMM parameters (Arribas-Gil et al. 2006). In our case, we will use a modified version of the Baum-Welch (Baum et al. 1970) Expectation Maximization algorithm for estimating Pair HMM parameters. This algorithm also uses a forward-backward procedure.

It is apparent from the tasks above that the standard algorithms for inference using Pair HMMs include: the Forward, Backward and Viterbi algorithms. These algorithms are based on the classic HMM algorithms and the reader is refered to Mackay (2004) for their derivation for Pair HMMs. Unfortunately, the implementation of these algorithms based on the classic HMMs requires that they be modified to suit a particular Pair HMM structure. In the following subsections, we provide pseudocode to illustrate the Forward, Backward, and Viterbi algorithms for the Pair HMM with distinct emission parameters and whose transition parameters are all distinct 4.4. We denote the source and target words by x and y respectively. We also use i and j to denote the indexes in the two sequences respectively. x_i denotes the i^{th} character in the source sequence and y_j denotes the j^{th} character in the target sequence. We use p_{x_i,y_j} to denote the probability of emitting the i^{th} source word character and j^{th} target word character when in a Pair HMM's substitution state. q_{x_i} is used to denote the probability of emitting the i^{th} source word character and an empty symbol in the deletion state (X) whereas q_{y_j} is used to denote the probability of emitting the empty symbol and the j^{th} target word character in the insertion state (Y).

a) Forward algorithm for the Pair HMM with distinct parameters

The Forward algorithm computes for all possible paths, the total probability of a pair of subsequences ($x_1, ..., x_i$ and $y_1, ..., y_j$) that have been emitted upto a hidden state k (M, X, Y). We use the variable $f^k(i, j)$ to denote this total probability, and $f^{\cdot}(i, j)$ to indicate the Forward probability associated with being in any of the states. The initialization, induction, and termination equations for the Forward algorithm specific to the Pair HMM of Figure 4.4 are as shown on the next page.

b) Backward algorithm for the Pair HMM with distinct parameters

In a manner similar to that of the Forward algorithm, the Backward algorithm computes the total probability for all possible paths of the subsequences starting from x_{i+1} and y_{j+1} up to the end, when the Pair HMM is in a state k. We use the variable $b^k(i, j)$ to denote the total Backward probability and $b^{\cdot}(i, j)$ to indicate the Backward probability associated with any of the states. We also use n and m to index the end of the source and target sequences respectively. The initialization, induction, and

The forward algorithm for a Pair HMM that uses distinct transition parameters

1. Initialization
   $f^M(0,0) = 1 - \delta_X - \delta_Y - \tau_M,\ f^X(0,0) = \delta_X,\ f^Y(0,0) = \delta_Y$
   All $f^{\cdot}(i,-1) = f^{\cdot}(-1,j) = 0$
2. Induction

for $0 \le i \le n,\ 0 \le j \le m,$ except $(0,0)$ **do**

$$f^M(i,j) = p_{x_i y_j}[(1 - \delta_X - \delta_Y - \tau_M)f^M(i-1,j-1) + (1 - \epsilon_X - \lambda_X - \tau_X)f^X(i-1,j-1) + (1 - \epsilon_Y - \lambda_Y - \tau_Y)f^Y(i-1,j-1)],$$
$$f^X(i,j) = q_{x_i}[\delta_X f^M(i-1,j) + \epsilon_X f^X(i-1,j) + \lambda_Y f^Y(i-1,j)],$$
$$f^Y(i,j) = q_{y_j}[\delta_Y f^M(i,j-1) + \lambda_X f^X(i,j-1) + \epsilon_Y f^Y(i,j-1)]$$

end for

3. Termination
   $P(O|\mu) = \tau_M f^M(n,m) + \tau_X f^X(n,m) + \tau_Y f^Y(n,m)$

The backward algorithm for a Pair HMM that uses distinct transition parameters

1. Initialization
   $b^M(n,m) = \tau_M, b^X(n,m) = \tau_X, b^Y(n,m) = \tau_Y$
2. Induction

$$b^M(i,j) = (1 - \delta_X - \delta_Y - \tau_M)p_{x_{i+1}y_{j+1}}b^M(i+1,j+1) + \delta_X q_{X_{i+1}}b^X(i+1,j) + \delta_Y q_{Y_{j+1}}b^Y(i,j+1),$$
$$b^X(i,j) = (1 - \epsilon_X - \lambda_X - \tau_X)p_{x_{i+1}y_{j+1}}b^M(i+1,j+1) + \epsilon_X q_{x_{i+1}}b^X(i+1,j) + \lambda_X q_{y_{j+1}}b^Y(i,j+1),$$
$$b^Y(i,j) = (1 - \epsilon_Y - \lambda_Y - \tau_Y)p_{x_{i+1}y_{j+1}}b^M(i+1,j+1) + \lambda_Y q_{x_{i+1}}b^X(i+1,j) + \epsilon_Y q_{y_{j+1}}b^Y(i,j+1).$$

3. Termination
   $P(O|\mu) = (1 - \delta_X - \delta_Y - \tau_M)b^M(0,0) + \delta_X b^X(0,0) + \delta_Y b^Y(0,0)$

termination equations of the Backward algorithm specific to the Pair HMM of Figure 4.4 are as shown above.

c) Viterbi algorithm for the Pair HMM with distinct parameters

The Viterbi algorithm is used to find the best alignment sequence(s) given a pair of observation sequences. We use the variable $v^{\cdot}(i,j)$ to denote the probability of emitting the aligned subsequences $x_1, ..., x_i$ and $y_1, ..., y_j$ by the Pair HMM with the sub alignment ending with (a) aligned pair x_i and y_j ($v^{\cdot}(i,j) = v^M(i,j)$), (b) x_i aligned to an empty string e ($v^{\cdot}(i,j) = v^X(i,j)$), and (c) y_j aligned to the empty

The Viterbi algorithm for a Pair HMM that uses distinct transition parameters

1. Initialization

   $v^M(0,0) = 1 - \delta_X - \delta_Y - \tau_M, \quad v^X(0,0) = \delta_X, \quad v^Y(0,0) = \delta_Y.$

   All $\quad v^{\cdot}(i,-1) = v^{\cdot}(-1,j) = 0$

2. Induction

for $0 \leq i \leq n$, $0 \leq j \leq m$ except $(0,0)$ **do**

$$v^M(i,j) = p_{x_i y_j} \max \left\{ \begin{array}{l} (1-\delta_X-\delta_Y-\tau_M)v^M(i-1,j-1) \\ (1-\epsilon_X-\lambda_X-\tau_X)v^X(i-1,j-1) \\ (1-\epsilon_Y-\lambda_Y-\tau_Y)v^Y(i-1,j-1) \end{array} \right\},$$

$$v^X(i,j) = q_{x_i} \max \left\{ \begin{array}{l} \delta_X v^M(i-1,j) \\ \epsilon_X v^X(i-1,j) \\ \lambda_Y v^Y(i-1,j) \end{array} \right\},$$

$$v^Y(i,j) = q_{y_j} \max \left\{ \begin{array}{l} \delta_Y v^M(i,j-1) \\ \lambda_X v^X(i,j-1) \\ \epsilon_Y v^Y(i,j-1) \end{array} \right\}$$

end for

3. Termination

   $P(H) = \max(\tau_M v^M(n,m), \tau_X v^X(n,m), \tau_Y v^Y(n,m)).$

string e ($v^{\cdot}(i,j) = v^Y(i,j)$). Using $v^{\cdot}(i,j)$ to indicate the probability in any of the states, the initialization, induction, and termination equations for the Viterbi algorithm specific to the Pair HMM of Figure 4.4 are as shown above.

d) Log-odds algorithms

The Forward, Backward, and Viterbi Pair HMM algorithms may be sufficient for computing transliteration similarity. Durbin et al. (1998) introduce another approach of using *log-odds* ratios to compute string similarity. A log-odds ratio is used to incorporate the likelihood of the random occurrence of a pair of observations in the computation of string similarity. Mackay and Kondrak (2005) show that using of the Viterbi log-odds algorithm results in significantly better cognate identification performance compared to using the standard Pair HMM algorithms. Since the requirements for computing transliteration similarity are similar to those for computing word similarity where the approach of *log-odds* ratios is reported to be more successful, we also propose to evaluate it for computing transliteration similarity.

The *log-odds* algorithms use a random Pair HMM (Figure 4.5) to represent the

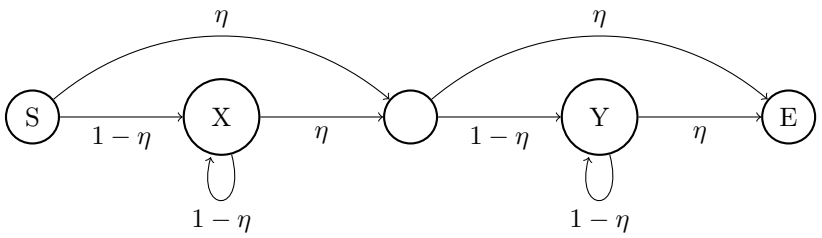

Figure 4.5: *Finite state representation of the random Pair HMM. This model uses only one transition probability (η) with deletion probabilities (r_{x_i}) and insertion probabilities (r_{y_j}). X and Y nodes respectively refer to deletion and insertion states, and S denotes a start state. The unlabeled node represents a silent state which does not emit any symbols but is used to gather inputs from S and X states. Adapted from Durbin et al (1998).*

likelihood of the random occurrence of a pair of strings in the source and target languages. The random Pair HMM does not have a match state since the source and target sequences are assumed to have no underlying relationship to each other. The random model in Figure 4.5 uses only one transition parameter[2] (η) with deletion probabilities (r_{x_i}) and insertion probabilities (r_{y_j}). In the following, we briefly review the Viterbi log-odds algorithm using Mackay and Kondrak's Pair HMM (Figure 4.2).

According to the random Pair HMM in Figure 4.5, the probability of the random occurrence of a pair of words can be computed as follows:

$$P(x, y|R) = \eta(1-\eta)^n \prod_{i=1}^{n} r_{x_i} \eta(1-\eta)^m \prod_{i=1}^{m} r_{y_j} = \eta^2(1-\eta)^{n+m} \prod_{i=1}^{n} r_{x_i} \prod_{i=1}^{m} r_{y_j}. \quad (4.5)$$

where x (having n characters) and y (having m characters) represent source and target words respectively. An additive model with resulting log-odds scores for emissions and transitions can be specified by combining emission scores and transition scores from the standard and random Pair HMMs. The following equations are used to merge the emission and transition scores for Mackay and Kondrak's Pair HMM (Figure 4.2) to get the standard terms necessary for sequence alignment following a dynamic programming methodology. In the equations below, we use $s(.,.)$ to denote the substitution score; $d(.)$ the *gap open* score; and $e(.)$ and $f(.)$ gap extension scores that correspond to transitions and emissions from the match state to the gap states, and between the gap states respectively.

[2]There should be no restriction on the number of transition parameters that can be used in the random Pair HMM. It should be possible to use two transition parameters, one associated with deletions, while the other with insertions. However it is the emission states X and Y that contribute more to the final random model probability.

$$s(x,y) = \log \frac{p_{xy}}{r_x r_y} + \log \frac{1-2\delta-\tau_M}{(1-\eta)^2}$$

$$d(x) = -\log \frac{q_x \delta(1-\epsilon-\lambda-\tau_{XY})}{r_x(1-\eta)(1-2\delta-\tau_M)}$$

$$e(x) = -\log \frac{q_x \epsilon}{r_x(1-\eta)}$$

$$f(x) = -\log \frac{q_x \lambda}{r_x(1-\eta)}$$

$$c = \log \frac{1-2\delta-\tau_M}{1-\epsilon-\lambda-\tau_{XY}} + \log(\tau_{XY}).$$

The Viterbi log-odds algorithm for Mackay and Kondrak's Pair HMM is as shown below.

The Viterbi log-odds algorithm for a Pair HMM that uses five transition parameters

---

1. Initialization

   $v^M(0,0) = -2\log(\eta), \quad v^X(0,0) = v^Y(0,0) = -\infty.$

   All $\quad v^{\cdot}(i,-1) = v^{\cdot}(-1,j) = 0$

2. Induction

for $0 \le i \le n$, $0 \le j \le m$ except $(0,0)$ **do**

$$v^M(i,j) = s_{x_i y_j} \max \left\{ \begin{array}{c} v^M(i-1,j-1) \\ v^X(i-1,j-1) \\ v^Y(i-1,j-1) \end{array} \right\},$$

$$v^X(i,j) = \max \left\{ \begin{array}{c} v^M(i-1,j) - d(x_i) \\ v^X(i-1,j) - e(x_i) \\ v^Y(i-1,j) - f(x_i) \end{array} \right\},$$

$$v^Y(i,j) = \max \left\{ \begin{array}{c} v^M(i,j-1) - d(y_j) \\ v^X(i,j-1) - e(y_j) \\ v^Y(i,j-1) - f(y_j) \end{array} \right\}.$$

end for

3. Termination

   $P(H) = \max(v^M(n,m) + \log(\tau_M), v^X(n,m) + c, v^Y(n,m) + c).$

The log-odds emission and transition score expressions, and the Viterbi log-odds algorithm for the other Pair HMMs are also derived based on the respective Pair

HMM parametric definitions. In a similar manner, we derive the Forward log-odds algorithms for all the Pair HMMs.

4.3.5 Pair HMMs – parameter estimation

A Pair HMM requires two main sets of parameters to compute transliteration similarity: transition and emission parameters. These parameters need to be estimated before a Pair HMM can be used for computing transliteration similarity. There exist different approaches for estimating these parameters; some approaches which are compared in literature include (Arribas-Gil et al. 2006): numerical maximization techniques, and Expectation Maximization (EM) algorithms with variants such as stochastic EM and stochastic approximation EM. For the transliteration detection task, we will adopt the EM-based Baum-Welch algorithm which has already found successful application in the cognate identification (Mackay and Kondrak 2005) and dialect comparison (Wieling et al. 2007) tasks.

Before we describe the Baum-Welch algorithm for the Pair HMMs, we begin by presenting the case for the classic HMMs. The main difference between estimating parameters for Pair HMMs and estimating parameters for classic HMMs is that for the Pair HMMs, we have to consider an extra dimension of observation sequences. Using common notation for the classic HMMs, we specify the transition probability from a state s_k to a state s_l at time t by the variable $\xi_t(k,l)$ which is formulated as:

$$\xi_t(k,l) = P(s_k, s_l|O^d, \mu) = \frac{P(s_k, s_l, O^d|\mu)}{P(O^d|\mu)}$$

where O^d is an observation sequence and μ is some HMM. Through expansion and simplification using forward ($f_t(k)$) and backward ($b_t(k)$) variables, it can be shown that

$$\xi_t(k,l) = \frac{f_t(k)P_{kl}e_l(O^d_{t+1})b_{t+1}(l)}{P(O|\mu)}$$

We also specify the probability of being in a state k at time t given the observation sequence by the variable $\gamma_t(k)$:

$$\gamma_t(k) = P(s_k|O^d, \mu) = \frac{P(s_k, O^d|\mu)}{P(O^d|\mu)} = \frac{f_t(k)b_t(k)}{\sum_{l=1}^{N} f_t(l)b_t(l)}$$

It can be seen that

$$\gamma_t(k) = \sum_{l=1}^{N} \xi_t(k,l) = \sum_{l=1}^{N} \frac{f_t(k)P_{kl}e_l(o_{t+1})b_{t+1}(l)}{P(O|\mu)}$$

$$= \frac{1}{P(O|\mu)} f_t(k)[\sum_{l=1}^{N} P_{kl} e_l(o_{t+1}) b_{t+1}(l)] = \frac{1}{P(O|\mu)} f_t(k) b_t(k)$$

If we sum over the time index for the two variables $\xi_t(k,l)$ and $\gamma_t(k)$, we get expectations (or counts) that can be used in re-estimating the parameters of an HMM using the following equations:

$$\pi_k = \text{expected number of times in state } k \text{ at time } t = 1 \qquad = \gamma_1(k)$$

$$P_{kl} = \frac{\text{expected number of transitions from state } k \text{ to state } l}{\text{expected number of transitions from state } k}$$

$$= \frac{\sum_{t=1}^{T-1} \xi_t(k,l)}{\sum_{t=1}^{T-1} \gamma_t(k)} = \sum_d \frac{1}{P(O^d)} \sum_t f_t(k) P_{kl} e_l(O^d_{t+1}) b_{t+1}(l)$$

$$e_k(v) = \frac{\text{expected number of times in state } k \text{ observing symbol } v}{\text{expected number of times in state } k}$$

$$= \frac{\sum_{t:o_t=v, 1\leq t\leq T} \gamma_t(l)}{\sum_{t=1}^{T} \gamma_t(l)} = \sum_d \frac{1}{P(O^d)} \sum_{t|O^d_t=v} f_t(k) b_t(k)$$

Now, for the Pair HMMs, we have to sum over the positions in each of the sequences, and over all possible sequences. Let h denote the index of the pair of sequences we are using, and let f and b denote the forward and backward variables respectively but this time with an extra dimension. We obtain the following equations for the transition and emission probabilities:

For a transition (k to l) ending in a substitution state we have

$$P_{kl} = \sum_h \frac{1}{P(O|\mu)} \sum_i \sum_j f^h_{(i,j)}(k) P_{kl} e_l(x^h_{i+1}, y^h_{j+1}) b^h_{(i+1,j+1)}(l) \tag{4.6}$$

For an emission in the substitution state we have

$$e_k(v^{(x_i,y_j)}) = \sum_h \frac{1}{P(O|\mu)} \sum_{i|x^h_i \in O^{xy}} \sum_{j|y^h_j \in O^{xy}} f^h_k(i,j) b^h_k(i,j) \tag{4.7}$$

The equations for the insertion and deletion states will have slightly different forms. In the insertion state, we need to match symbol y_j only. Therefore in the equation

for estimating transition probability, we change the index for only one sequence of the pair and we use the emission probability for a symbol from one string against a gap. The probability estimations for the gap states are as follows:

For the deletion state X:

$$P_{kl} = \sum_h \frac{1}{P(O|\mu)} \sum_i \sum_j f^h_{(i,j)}(k) P_{kl} e_l(x^h_{i+1}) b^h_{(i+1,j)}(l) \tag{4.8}$$

$$e_k(O^{xy}) = \sum_h \frac{1}{P(O|\mu)} \sum_{i|x^h_i \in O^{xy}} \sum_j f^h_k(i,j) b^h_k(i,j) \tag{4.9}$$

For the insertion state Y:

$$P_{kl} = \sum_h \frac{1}{P(O|\mu)} \sum_i \sum_j f^h_{(i,j)}(k) P_{kl} e_l(y^h_{j+1}) b^h_{(i,j+1)}(l) \tag{4.10}$$

$$e_k(O^{xy}) = \sum_h \frac{1}{P(O|\mu)} \sum_i \sum_{j|y^h_j \in O^{xy}} f^h_k(i,j) b^h_i(i,j) \tag{4.11}$$

These expressions enable the use of a forward-backward procedure to learn all the transition and emission parameters of a Pair HMM.

4.4 Transliteration detection experiments using geographic names data

4.4.1 Data

For the preliminary transliteration detection experiments, we obtained geographic name pairs from the Geonames[3] database. This database provides a collection of geographic names and how they are alternately represented using different languages. We consider only four language pairs: English-Dutch, English-French, English-German, and English-Russian. As this list of language pairs shows, we also include in our investigation the case where the source and target language use the same writing system. We propose this as an alternative solution for building bilingual named entity lexicons where names are spelled differently across the languages even for the case of the same writing system.

After extracting the raw collection of geographic name pairs from the Geonames database, we manually checked each dataset (for each language pair) to remove any noisy and / or irrelevant entities. For the language pairs where the same writing

[3] `http://www.geonames.org`

system is used, we filtered out name pairs where the names have the same spelling. This resulted in much smaller sizes of the datasets for these language pairs. We also found frequent use of diacritics and accents in the datasets. We did not normalize any diactrics to the standard Latin alphabet of 26 characters. The presence of the unusual characters for these language pairs partly justifies our assumption to analyse the data in the context of transliteration since the diacritics and accents convey different pronunciations. Table 4.1 gives a summary of the dataset sizes after pre-processing.

Language pair	Total no. of NE pairs
English-Russian	25104
English-Dutch	514
English-French	784
English-Germany	856

Table 4.1: *Total size of cross language name pairs that were obtained from the geonames database.*

4.4.2 Evaluation setup and results

In this section, we evaluate two Pair HMM settings in an experimental transliteration detection task. In the first setting, we specify Mackay and Kondrak's Pair HMM to use the same alphabet for the source and target observation sequences. We refer to this Pair HMM setting as PHMM1. In the second setting, we apply Mackay and and Kondrak's Pair HMM with distinct insertion and deletion parameters corresponding to two character vocabularies (one vocabulary for generating source language characters whereas the other is for generating target language characters). We refer to the second Pair HMM setting as PHMM2. Table 4.2 shows the vocabulary sizes that were obtained from data for the four language pairs using the two Pair HMM settings.

Language pair	PHMM1	PHMM2	
English-French	66	57 English	52 French
English-Russian	127	91 English	61 Russian
English-Dutch	62	56 English	47 Dutch
English-Germany	33	32 English	32 German

Table 4.2: *Total sizes of the alphabets for PHMM1 and PHMM2 models.*

The aim of investigating the two Pair HMM settings at this stage is to determine whether the assumption of generating source and target observations based on a single alphabet is sufficient for transliteration similarity estimation. We also evaluate the use of most of the Pair HMM algorithms presented in Section 4.3.4 for computing transliteration similarity.

In the previous subsection, the small size of the experimental datasets limits our evaluation of the Pair HMMs. In this chapter, we follow two approaches to address this limitation. In the first approach, we propose to use the transliteration similarity likelihood that each model assigns to correct named entity matches in the transliteration corpus. We therefore apply an information theoretic measure known as *corpus cross entropy* (CCE) which is computed based on the likelihoods. CCE can be used to evaluate the models without the need for a test corpus. Usually, CCE is used to give an initial idea about how well the models approximate the true representation of data in the corpus. We provide such an evaluation for the English-Russian dataset. However, in order to evaluate the models for identifying transliterations, we need to use standard evaluation metrics. For the second approach, we perform *K-fold cross validation* where we use the usual standard evaluation metrics of *accuracy* and *mean reciprocal rank* (MRR).

Starting with the first approach, we introduce the concept of cross entropy in the context of comparing the models we have proposed to compute transliteration similarity. It is important to note that we provide a rather detailed introduction since there is scarcely a discussion on the use of cross entropy for comparing transliteration models in the machine transliteration literature. Later, we present our application of *corpus cross entropy* that we use to compare the two Pair HMM settings including the inference algorithms.

a) Corpus cross entropy

Entropy in information-theoretic terms is used to quantify the uncertainty associated with a random variable. In the context of transliteration, the random variables can be perceived to range over characters or words. If we have a set of events whose probabilities of occurrence are $P(x_1), P(x_2), ..., P(x_N)$, we may want to know the extent of uncertainty for the events. Such a measure, denoted by $\mathrm{H}(x)$ is defined by Shannon (1948) to have the following properties: (1) $\mathrm{H}(x)$ should be continuous in $P(x_i)$. (2) If the $P(x_i)$ are equal, $P(x_i) = \frac{1}{N}$, then H should be a monotonic increasing function. (3) if a choice is broken down into two successive choices, the original $\mathrm{H}(x)$ should be the weighted sum of the individual values of $\mathrm{H}(x)$. The only $\mathrm{H}(x)$, that satisfies these three assumptions is specified by Equation 4.12 (Shannon 1948) where K is a positive constant. $\mathrm{H}(x)$ is referred to as the entropy of the

probability distribution over the events.

$$\mathrm{H}(x) = -\mathrm{K}\sum_{i=1}^{\mathrm{N}} P(x_i)\log P(x_i) \tag{4.12}$$

The choice of the logarithmic base corresponds to the preferred unit for measuring information. For a character sequence $S = s_1, s_2, ..., s_{\mathrm{N}}$, we can specify the entropy using Equation 4.13.

$$\mathrm{H}(s_1, s_2, ..., s_{\mathrm{N}}) = -\sum_{S_1^{\mathrm{N}} \in A} P(S_1^{\mathrm{N}})\log P(S_1^{\mathrm{N}}) \tag{4.13}$$

where A is the vocabulary of characters for a given language.

The concept of *cross entropy* comes into play for cases where we use a model distribution (which is a close approximation to P) instead of the true distribution P. Specifically, *cross entropy* measures the amount of information needed to represent a test event using the model distribution. The interest usually is to know how well the model distribution approximates p. Equation 4.14 below defines the *cross entropy* between the model distribution $m(x_i)$ and the true distribution $P(x_i)$:

$$\mathrm{H}(p, m) = -\sum\nolimits_i P(x_i)\log m(x_i) \tag{4.14}$$

One resulting property for *cross entropy* is that it is an upper bound on the *true entropy* $\mathrm{H}(P)$. That is, given a model m, $\mathrm{H}(P) \leq \mathrm{H}(P, m)$. If $P = m$, *cross entropy* is said to be at a minimum and $\mathrm{H}(P, m) = \mathrm{H}(P)$. The closer the *cross entropy* $\mathrm{H}(P, m)$ is to the true entropy $\mathrm{H}(P)$, the better m becomes an approximation of p. We can therefore use *cross entropy* to compare approximate models. Between two models m_1 and m_2, the more accurate model should be the one with the lower *cross entropy*. Here, lower *cross entropy* means that the result is less surprising since we need very little information to represent the test event. *Cross entropy* is used to derive the *Perplexity* measure which is used in many fields to measure how well a given model fits the data: $\mathit{Perplexity} = 2^{\mathit{Cross\ entropy}}$.

We formulate the *cross entropy* involving a Pair HMM that models the similarity between a pair of observation sequences as shown in Equation 4.15 below:

$$\mathrm{H}(P, m) = -\sum_{s \in A_1, t \in A_2} P(s_1 : t_1, ..., s_T : t_T)\log m(s_1 : t_1, ..., s_T : t_T) \tag{4.15}$$

In this case, we draw the pairs of observations according to the true probability distribution P, but sum the log of their probabilities according to m. If we assume that all the tokens (events) in the corpus are distinct, then the *true probability distribution* is simply a *uniform distribution*. Along a different line of reasoning, we can simply

use the *log probability* that the model assigns to the tokes in the corpus for evaluation. This *log probability* is referred to as the *corpus cross entropy*. Given a corpus $\mathcal{C}$ of size N consisting of tokens $c_1, c_2, ..., c_N$, the *log probability* of a model m on this corpus is defined by the following equation:

$$\mathrm{H}_{\mathcal{C}}(m) = -\left(\frac{1}{\mathrm{N}}\right) \times \sum\nolimits_i \log m(c_i) \tag{4.16}$$

where the summation is done over the tokens in the corpus. It can be proven that as N tends to ∞, the *corpus cross entropy* becomes the *cross entropy* for the *true distribution*. To prove the equivalence of the *corpus cross entropy* with the *true entropy*, it must be assumed that the corpus has a stationary distribution. The proof depends on the fact that the maximum likelihood estimate approaches the true probability distribution as the size of the corpus tends to ∞.

According to Manning and Schutze (1999), it is not exactly correct to use use the result for corpus cross entropy in NLP applications because the stationary distribution assumption is clearly wrong for natural languages. Nonetheless, for a given corpus, we can assume that a language is near enough unchanging and this can be considered as an acceptable approximation to truth (Askari, 2006).

In our computation for CCE, we regard each named entity (NE) pair (s^i, t^i) as a token c_i. We formulate the CCE of a given Pair HMM m (CCE_m) on a transliteration corpus as follows:

$$\mathrm{CCE}_m = -\left(\frac{1}{\mathrm{N}}\right) \times \sum_{i=1}^{N} \log(m(s^i, t^i)) \tag{4.17}$$

where $m(s^i, t^i)$ is the probability the Pair HMM m assigns to a source and target language NE pair (s^i, t^i).

CCE results on English-Russian geographic names corpus

Here, we divided the English-Russian dataset into two where 90% (22594) of the name pairs were used as training data and the rest as test data. The training data is used to estimate the parameters of the Pair HMMs using the Baum-Welch algorithm. The scoring algorithms for both Pair HMMs (PHMM1 and PHMM2) are then applied to compute a *log probability score* for each name pair entry in the test set using the parameters of the trained Pair HMMs. We then use Equation 4.17 to compute the CCE per algorithm for each Pair HMM. Table 4.3 shows the resulting CCE for the Pair HMM algorithms on the whole English-Russian transliteration corpus.

It is clear from Table 4.3 that the use of distinct emission parameters based on two distinct alphabets results in a relatively lower cross entropy (hence less uncertainty)

Pair HMM algorithm	CCE	
	PHMM1	PHMM2
Viterbi	34.085	23.907
Forward	33.970	21.201
Viterbi log-odds (identical)	220.468	110.633
Viterbi log-odds (distinct)	76.199	48.229
Forward log-odds (identical)	107.927	107.927
Forward log-odds (distinct)	63.353	45.248

Table 4.3: *CCE for the Pair HMM algorithms on our English-Russian Geonames transliteration corpus.*

compared to the case where one alphabet is used. It is important to note that we can only use CCE to compare corresponding algorithms of the two Pair HMMs. We can not use CCE to evaluate the base Pair HMM algorithms with the log-odds algorithms since a base Pair HMM algorithm is also involved in the computation of the log-odds score. This would always result in a higher value of CCE for the log-odds algorithm compared to that for the base Pair HMM algorithm.

We can also compare the two models by looking at the rate of change of CCE as the size of the test set increases. The graphs in Figure 4.6 show the variation of CCE with the size of the English-Russian Geonames transliteration test set for the different Pair HMM scoring algorithms.

The differences in the curves for the corresponding algorithms in Figure 4.6 also suggest that using two alphabets (PHMM2) generally makes Pair HMMs better at modeling transliteration similarity than when they use one alphabet. However, before we make a final conclusion, we evaluate the models in a standard transliteration detection setting using two traditional metrics: *transliteration detection accuracy* and *mean reciprocal rank* (MRR). In the following subsection, we define the two metrics.

b) Transliteration detection accuracy and mean reciprocal rank

Transliteration detection accuracy measures the proportion of target named entities that are correctly detected as transliteration matches of the respective source named entities.

$$\text{Accuracy} = \frac{\text{number of correctly identified target named entities}}{\text{total number of test named entities identified}}$$

Transliteration detection accuracy can be computed at a particular cutoff rank. TOP-1 accuracy (or sometimes referred to as *precision at rank 1*) indicates the proportion

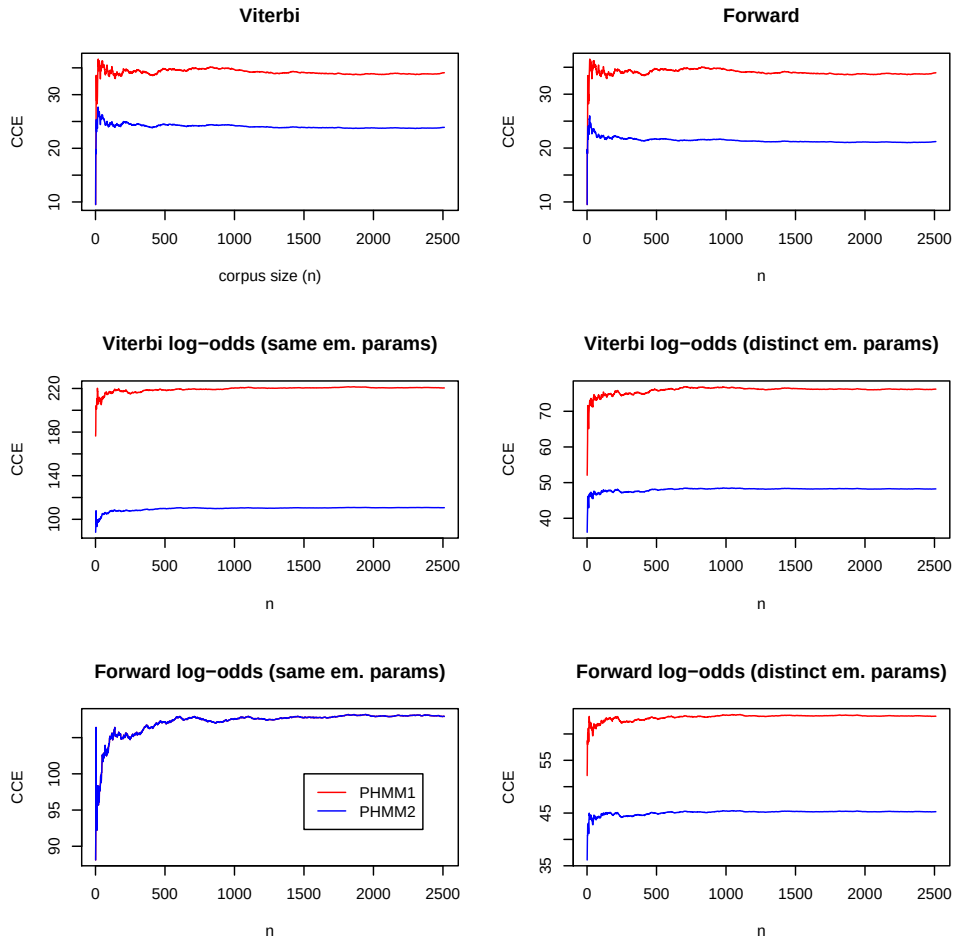

Figure 4.6: *Variation of CCE with the English-Russian test set size for different Pair HMM scoring algorithms.*

of named entities in the test set for which the correctly detected target named entity was the first to be returned by the transliteration system. Likewise TOP-5 accuracy indicates the proportion of test named entities for which the correct transliteration was returned within the first five candidate transliterations. If more than one transliteration is available for a source word in the test data, we need to take the variant transliterations into account. A modified version of the transliteration detection met-

ric will be used in Chapter 7 for this purpose. Equation 4.18 below takes into account the case where we have more than one transliteration for a source word.

$$\text{Accuracy } = \frac{1}{\text{N}} \sum_{i=1}^{\text{N}} 1 \text{ if } \exists\ r_{i,j} = c_{i,1};\ 0 \text{ otherwise} \tag{4.18}$$

where $r_{i,j}$ is the j^{th} reference transliteration for the i^{th} source word in test data and $c_{i,1}$ is the first candidate transliteration returned by the transliteration system. N is the size of the test dataset.

For the transliteration detection experiments in this chapter, we assume that there is only one transliteration for each source word. We also evaluate the methods using only TOP-1 accuracy.

We also use MRR, which is mainly used in information retrieval to evaluate the ranked list of documents returned by a search system. In the context of transliteration detection, MRR is the average of reciprocal ranks associated with the NEs in the test set. A reciprocal rank is the reciprocal of the rank at which the correct target transliteration was identified for a given source NE.

$$\text{MRR} = \frac{1}{\text{N}} \sum_{i=1}^{\text{N}} \frac{1}{\text{R}_i} \tag{4.19}$$

where R_i is the rank of the correct transliteration in the returned list of candidate transliterations for the i^{th} test NE. N is the total number of NEs in the test set.

When using these metrics for the preliminary set of experiments, we perform *stratified K-fold cross-validation* for evaluation. We use Equations 4.18 and 4.19 to compute the accuracy and MRR respectively in each fold. We then compute the cross validation accuracy (CVA) and cross validation MRR (CVMRR) by averaging the K individual accuracy and MRR values from each of the K folds.

$$\text{CVA} = \frac{1}{K} \sum_{k=1}^{K} A_k. \tag{4.20}$$

where A_k is the accuracy of a model on the k^{th} test data set (see Equation 4.18) while K is the total number of test datasets for cross-validation. Likewise,

$$\text{CVMRR} = \frac{1}{K} \sum_{k=1}^{K} \text{MRR}_k \tag{4.21}$$

where MRR_k is the MRR of a model on the k^{th} test data set (see Equation 4.19).

Transliteration detection accuracy and MRR results

Here, we evaluate the models on all the four language pairs for which we perform stratified 10-fold cross validation. During training, a Pair HMM Baum-Welch algorithm is used to estimate the emission and transition parameters of the corresponding Pair HMM. Table 4.4 shows the average number of iterations that were required for the Baum-Welch algorithm to converge for three of the four language pairs.

Language pair	avg. no. of iterations	
	PHMM1	PHMM2
English-Russian	802	702
English-Dutch	896	839
English-Germany	418	379

Table 4.4: *Average number of iterations required by the Baum-Welch algorithm to converge to local maximum for PHMM1 and PHMM2.*

Table 4.4 suggests that Baum-Welch algorithm converges faster to a local optimum for the case of PHMM2 than the case for PHMM1 on all the language pairs. Table 4.5 shows the CVA and CVMRR results from the use of the scoring algorithms for the two Pair HMMs on the English-French dataset.

Pair HMM algorithm	CVA		CVMRR	
	PHMM1	PHMM2	PHMM1	PHMM2
Viterbi	0.836	0.824	0.858	0.850
Forward	0.834	0.827	0.857	0.852
Vit. log-odds (identical)	0.675	0.682	0.751	0.752
Vit. log-odds (distinct)	0.835	0.844	0.861	0.866
Forward log-odds (identical)	0.676	0.677	0.751	0.751
Forward log-odds (distinct)	**0.845**	**0.859**	**0.866**	**0.875**

Table 4.5: *CVA and CVMRR transliteration detection results for different Pair HMM algorithms on English-French data. Values in* ***bold*** *indicate the best result.*

The results in Table 4.5 suggest that if the standard Pair HMM algorithms are used for computing transliteration similarity, there is no gain in using two separate alphabets between English and French. Although all the log-odds versions of the standard algorithms for PHMM2 achieve a slight improvement in transliteration detection quality compared to those for PHMM1, the differences in CVA and CVMRR are not significant. Table 4.5 also shows that the log-odds algorithms that use dis-

tinct emission parameters result in better transliteration detection quality than the standard algorithms for both Pair HMMs.

Table 4.6 shows the CVA and CVMRR results from the application of PHMM1 and PHMM2 on the English-Russian dataset.

Pair HMM algorithm	CVA		CVMRR	
	PHMM1	PHMM2	PHMM1	PHMM2
Viterbi	0.759	0.824	0.788	0.834
Forward	0.759	0.830	0.788	0.838
Vit. log-odds (identical)	0.619	0.722	0.680	0.770
Vit. log-odds (distinct)	0.696	0.824	0.740	0.835
Forward log-odds (identical)	0.749	0.750	0.790	0.790
Forward log-odds (distinct)	**0.831**	**0.833**	**0.840**	**0.841**

Table 4.6: *CVA and CVMRR transliteration detection results for two Pair HMMs and different scoring algorithms on English-Russian Geonames data.*

In Table 4.6, we see a general improvement in transliteration detection quality from PHMM1 to PHMM2. Specifically, we see a considerable improvement in CVA and CVMRR for the standard Pair HMM algorithms. Again, The Forward log-odds algorithm that uses distinct emission parameters for the random and standard Pair HMMs results in relatively high transliteration detection quality compared to other scoring algorithms although with a slight difference compared to the standard Pair HMM algorithms in PHMM2. There is also a bigger difference between PHMM1 and PHMM2 in using the Forward log-odds algorithm as compared to the case for English-French above.

Table 4.7 shows the CVA and CVMRR results from the application of PHMM1 and PHMM2 on the English-Dutch dataset. The results in Table 4.7 suggest that the Viterbi log-odds algorithm achieves a relatively higher transliteration detection accuracy compared to the other Pair HMM algorithms although just slightly higher as compared to the forward log-odds algorithm. Table 4.7 results also suggest that not all algorithms result in improved transliteration detection quality when used for PHMM2. Generally the differences in CVA and CVMRR between the corresponding PHMM1 and PHMM2 algorithms are are so small.

Table 4.8 shows the CVA and CVMRR results from the application of PHMM1 and PHMM2 on the English-German dataset. Just like in the results for the English-Dutch dataset, Table 4.8 shows that there is hardly any difference in using PHMM1 compared to using PHMM2. Again the log-odds algorithms that use distinct emission parameters achieve higher transliteration detection accuracy compared to the base algorithms.

Pair HMM algorithm	CVA		CVMRR	
	PHMM1	PHMM2	PHMM1	PHMM2
Viterbi	0.804	0.796	0.836	0.829
Forward	0.806	0.808	0.836	0.838
Vit. log-odds (identical)	0.781	0.779	0.821	0.817
Vit. log-odds (distinct)	**0.824**	**0.826**	**0.846**	0.846
Forward log-odds (identical)	0.783	0.783	0.821	0.821
Forward log-odds (distinct)	0.820	0.824	0.845	**0.847**

Table 4.7: *CVA and CVMRR transliteration detection results for two Pair HMMs and different scoring algorithms on English-Dutch Geonames data.*

Pair HMM algorithm	CVA		CVMRR	
	PHMM1	PHMM2	PHMM1	PHMM2
Viterbi	0.893	0.895	0.920	0.919
Forward	0.898	0.896	0.922	0.922
Vit. log-odds (identical)	0.802	0.805	0.859	0.860
Vit. log-odds (distinct)	0.829	**0.920**	**0.938**	**0.938**
Forward log-odds (identical)	0.808	0.808	0.865	0.865
Forward log-odds (distinct)	**0.914**	0.911	0.935	0.934

Table 4.8: *CVA and CVMRR transliteration detection results for two Pair HMMs and different scoring algorithms on English-German Geonames data.*

c) Discussion

From the results on the four language pairs (that is English-French, English Russian, English-Dutch and English-German), we saw some variations in transliteration detection quality on using the different Pair HMM algorithms. The log-odds algorithms (that use distinct emission parameters between the random and standard Pair HMM) consistently performed well on all datasets based on the CVA and CVMRR metrics. The use of an information theoretic metric (*corpus cross entropy*) also suggested that PHMM2 algorithms are better approximators of transliteration similarity than PHMM1 algorithms on the English-Russian dataset. The results also suggest that some algorithms seem to be insensitive to changes in model settings, for example, the Forward log-odds algorithm resulted in almost the same transliteration detection quality under the two Pair HMM settings while the transliteration detection quality

for the Forward and Viterbi Pair HMM algorithms changed considerably. The results from the experimental transliteration detection task also differ from those reported in previous work. In the cognate recognition task in (Mackay and Kondrak 2005) and the Dutch dialect comparison task (Wieling et al. 2007), the Viterbi log-odds algorithm is reported to have performed better than all the other algorithms on nine language pairs that were used. This was also the case for the individual language pair of English-French in the cognate identification task. However, for English-French and English-Russian, the Forward log-odds algorithm performed consistently better. This seems to suggest that the properties associated with transliteration data may indeed differ from those of the datasets in tasks similar to transliteration detection thus necessitating a check on various model settings. We specifically see bigger differences in CVA and CVMRR between PHMM1 and PHMM2 for the English-Russian dataset where the languages indeed use different writing systems than the case for the other language pairs where the languages use the same writing system.

4.5 Experiments using NEWS 2009 and 2010 shared task data

4.5.1 Data

For the transliteration detection experiments in this section, we use manually verified transliteration data from the NEWS 2009 (Li et al. 2009) and 2010 (Li et al. 2010) shared tasks on transliteration generation. The transliteration data has been made available as standard data for evaluating machine transliteration systems. We evaluate the transliteration detection methods on seven language pairs: English-Bengali, English-Chinese, English-Hindi, English-Kannada, English-Russian, English-Tamil and English-Thai. For some language pairs, the respective shared task datasets did not need any further processing before being used. However, for some language pairs, the datasets needed some pre-processing which we describe in the following.

NE pairs that had spaces in them and of which the total size of the constituent names was equal on the source and target side, were split into single NE pairs. We assumed a monotonic ordering on the space-separated NEs and matched single names by their corresponding position. It was common to find the use of a comma in the space-separated NEs, especially in person names (for example 'WATSON, JOHN B.' in English and 'วัตสัน, จอห์น บี.' in Thai). The comma and the ordering in which the names were written was always the same on both the source and target side. From the English-Thai example above, the name 'วัตสัน' in the first position in the Thai NE is indeed a true Thai transliteration of the name 'WATSON' which is also in the first position in the English NE, and the same is true for the two remaining

strings in the respective positions of the complete NEs. We could as well have left out the space-separated NE pairs but a manual verification[4] on the split-matched NEs showed that almost all of the mappings were correct. We also removed any NE pairs that had unnecesary representations in them including: numerical representations, abbreviations, and person titles (such as Mrs., Mr., Dr., etc.). Table 4.9 shows the total number of single NE pairs that were used per language pair.

Language pair		Total size	Training	Testing
English-Bengali	(En-Be)	14255	12814	1441
English-Chinese	(En-Ch)	37228	33505	3723
English-Hindi	(En-Hi)	16059	14486	1573
English-Kannada	(En-Ka)	14368	12940	1428
English-Russian	(En-Ru)	7840	7056	784
English-Tamil	(En-Ta)	14622	13164	1458
English-Thai	(En-Th)	29050	26126	2924

Table 4.9: *Number of named entities from NEWS 2010 shared task standard transliteration data for Training and Testing per language pair.*

The English-Chinese datasets were first transformed into a Pinyin representation ignoring tones. A preliminary run on the original Chinese orthographic representation using all the Pair HMM variants proposed in this chapter resulted in very low transliteration detection accuracy. However, based on recent literature (Jiampojamarn et al. 2010, Zhao et al. 2007, Zhou 2009), a plausible approach for Chinese involves transforming the characters into Latin character representation using a Romanization system such as Pinyin to simplify transliteration analysis. For example, the Chinese characters in 彼得 would be transformed as follows using the Pinyin Romanization system: '彼→bi' and '得→de' (assuming we ignore tones). With such a transformation, we have the option of using the same vocabulary to represent the English and Chinese strings when applying the Pair HMMs. However, we specify the models to use distinct alphabets corresponding to the different writing systems. In English-Chinese transliteration detection, it is also sufficient to use the Romanized form for analysis, but in a transliteration generation task where the target string should be a Chinese string, we would need to transform[5] the Romanized representation into Chinese characters. Finally, for some language pairs, the named entities

[4] Verification in this case was possible since this the total size of this class of space separated NE pairs did not exceed 500 on all language pairs.

[5] This would require another step in the process of transliteration generation, that is, the segmentation of the string in the source language to enable mapping to a Romanized form for converting to Chinese characters.

were already sorted alphabetically on the English side and had to be randomized before dividing the whole set into train and test sets.

4.5.2 Evaluation setup and results

For this set of experiments, we report on results for only one held-out[6] test set for each language pair; we therefore evaluate the models using *transliteration detection accuracy* (Equation 4.18) and *Mean Reciprocal Rank* (MRR) (Equation 4.19). In this section, we evaluate the Pair HMMs against a standard baseline of using pair n-gram information.

a) Baseline – Pair n-gram models

We use as a baseline, a method that relies on the correspondence of source and target n-grams for estimating transliteration similarity. In its own right, this baseline method has a large model space for which we can derive various probabilistic and non-probabilistic similarity estimation schemes.

Consider a pair tri-gram case between English and Russian. To describe the method, we use the following name pair "Peter" (English) and "пётр" (Russian) which we assume exists in training data. We would like to elicit tri-gram relationships using the name pair by identifying tri-gram correspondences at the same position in the source and target. To ensure that we do not miss out on any characters in the source or target during the tri-gram correspondence identification process, we introduce a 'dummy' character denoted by # which we add at the start and end of each name as follows:

Peter → {##P, #Pe, Pet, ete, ter, er#, r##}
пётр → {##п, #пё, пёт, ётр, тр#, р##}

First, we move in the forward direction, while storing information about the tri-gram correspondences. For the name pair example above, we obtain the following correspondences in the forward direction:

$$\text{Tri-gram pairs}_{\text{forward}} = \{\text{##P-##п, \#Pe-\#пё, Pet-пёт, ete-ётр, ter-тр\#, er\#-р\#\#}\}$$

After the forward pass, we move in the backward direction resulting in the following tri-gram correspondences:

$$\text{Tri-gram pairs}_{\text{backward}} = \{\text{r\#\#-р\#\#, er\#-тр\#, ter-ётр, ete-пёт, Pet-\#пё, \#Pe-\#\#п}\}$$

[6] We also performed 10-fold cross validation but there was hardly any error of margin to using only one held-out test set. We report on results for only one held-out test set since we shall use them for comparison purposes with several other models in the next chapter.

The transliteration similarity scheme we use in this approach is based on whether or not a tri-gram correspondence associated with a candidate pair in the test set matches any of the tri-gram correspondences identified from the name pairs in the train set. If it does match, then the transliteration similarity score for the candidate pair is incremented by 1 otherwise we evaluate the next trigram pair until the end. Apart from tri-gram relationships, we also explore bi-gram, 4-gram, 5-gram, and 6-gram relationships in a similar manner.

We have applied this n-gram approach to transliteration data for the seven language pairs. Table 4.10 shows the *transliteration detection accuracy* and MRR for the different pair n-gram models. As can be seen in Table 4.10, the accuracy and MRR for the pair tri-gram model and the other higher order pair n-gram models are already high for five of the language pairs. As mentioned before, the pair n-gram approach can be associated with various scoring schemes. In a stratified 10-fold cross validation transliteration detection experiment, we found that the use of the scoring scheme described in the last paragraph led to considerably better transliteration detection accuracy and MRR than the case when we use direct probabilities computed for the n-gram correspondences in the train set. Table 4.10 also shows that apart from the case for English-Chinese, the pair tri-gram model is generally better at modeling transliteration similarity in comparison to the other pair n-gram models.

Model	Language Pair						
	En-Be	En-Ch	En-Hi	En-Ka	En-Ru	En-Ta	En-Th
	accuracy						
bigram	0.198	0.017	0.393	0.283	0.837	0.138	0.142
trigram	**0.804**	0.524	**0.813**	**0.763**	**0.944**	**0.775**	0.590
4-gram	0.767	**0.678**	0.809	0.738	0.922	0.762	**0.623**
5-gram	0.724	0.632	0.802	0.708	0.903	0.725	0.588
6-gram	0.728	0.614	0.806	0.710	0.903	0.705	0.549
	MRR						
bigram	0.252	0.026	0.453	0.341	0.866	0.188	0.186
trigram	**0.844**	0.604	**0.850**	**0.806**	**0.956**	**0.824**	0.659
4-gram	0.811	**0.736**	0.843	0.778	0.937	0.804	**0.685**
5-gram	0.771	0.683	0.837	0.751	0.920	0.768	0.645
6-gram	0.785	0.675	0.845	0.755	0.925	0.759	0.609

Table 4.10: *Transliteration detection accuracy and MRR for different pair n-gram models on seven language-pairs. Bold values indicate best results for each language pair.*

b) Pair HMM results

For the set of results reported in this section, we estimated each Pair HMM's parameters using the Baum-Welch EM algorithm starting with uniform initial probability distributions for starting, emission, and transition parameters. We then used the transliteration similarity estimates computed by six scoring algorithms to detect transliteration pairs from bilingual lists of candidate source and target NEs. We evaluate transliteration detection quality from using each Pair HMM scoring algorithm by computing transliteration detection accuracy and MRR. Tables 4.11 to 4.17 show the transliteration detection accuracy and MRR results for different scoring algorithms (for each Pair HMM variant) on test data for the seven language pairs. In the Tables, PHMM0 denotes the Pair HMM variant which is specified and trained with only emission parameters and no transition parameters between edit states; PHMM3 denotes the Pair HMM which is specified and trained to use three transition parameters between edit states (Figure 4.3); PHMM5 denotes the Pair HMM which uses five transition parameters between the edit states and distinct emission parameters in all the edit states (Figure 4.2); and PHMM9 denotes the Pair HMM that uses nine distinct transition parameters between the edit states in addition to distinct emission parameters in each edit state (Figure 4.4). The accuracy and MRR values in **bold** indicate the best result for a given Pair HMM and those in ***bold and italicized*** indicate the overall best result for that language pair. As shown in almost all tables, the Forward log-odds algorithm (where the random model's insertion and deletion parameters are based on character frequencies from the respective training data) for each Pair HMM variant consistently and in most cases outperforms the other algorithms on all transliteration data for the seven language pairs. On the other hand, we see that the Forward log-odds and Viterbi log-odds (that use the same insertion and deletion parameters for the standard models and the random model) consistently result in poor transliteration detection quality for all language pairs. We also see that although there is a general improvement in accuracy and MRR from the lack of Pair HMM transition parameters (PHMM0) to using transition parameters (PHMM3, PHMM5, PHMM9), an increase in the number of transition parameters does not guarantee improvements in transliteration detection quality. This is the case with PHMM9 which despite having the best accuracy and MRR from the Forward and Viterbi algorithms for almost all language pairs, results in lower transliteration detection (TD) quality on the best performing Forward log-odds algorithm. It is also surprising to see that for atleast four of the language pairs (En-Be, En-Ch, En-Hi, and En-Th), the Forward log-odds algorithm for PHMM3 results in the best overall accuracy and MRR whereas the other algorithms lead to poor TD quality in comparison to other Pair HMM variants. For the English-Russian dataset, we expected

Pair HMM algorithm	PHMM0	PHMM3	PHMM5	PHMM9
	accuracy			
Viterbi	0.717	0.579	0.776	0.795
Forward	0.822	0.731	0.869	0.862
Vit. log-odds (identical)	0.462	0.523	0.524	0.552
Vit. log-odds (distinct)	0.877	0.754	0.783	0.832
Forward log-odds (identical)	0.501	0.614	0.571	0.588
Forward log-odds (distinct)	**0.902**	***0.932***	**0.873**	**0.912**
Pair tri-gram baseline	0.804			
	MRR			
Viterbi	0.773	0.655	0.832	0.844
Forward	0.883	0.811	0.906	0.897
Vit. log-odds (identical)	0.583	0.631	0.625	0.659
Vit. log-odds (distinct)	0.918	0.803	0.835	0.878
Forward log-odds (identical)	0.621	0.737	0.678	0.704
Forward log-odds (distinct)	**0.936**	***0.950***	**0.911**	**0.937**
Pair tri-gram baseline	0.844			

Table 4.11: *English-Bengali transliteration detection accuracy and MRR for Pair HMMs.*

Pair HMM algorithm	PHMM0	PHMM3	PHMM5	PHMM9
	accuracy			
Viterbi	0.205	0.111	0.418	0.426
Forward	0.293	0.236	0.482	0.487
Vit. log-odds (identical)	0.050	0.080	0.132	0.132
Vit. log-odds (distinct)	0.180	0.237	0.537	0.537
Forward log-odds (identical)	0.086	0.129	0.145	0.144
Forward log-odds (distinct)	**0.467**	***0.684***	**0.666**	**0.661**
Pair tri-gram baseline	0.524			
	MRR			
Viterbi	0.286	0.163	0.533	0.538
Forward	0.411	0.358	0.611	0.615
Vit. log-odds (identical)	0.088	0.126	0.209	0.209
Vit. log-odds (distinct)	0.233	0.296	0.634	0.632
Forward log-odds (identical)	0.150	0.205	0.228	0.226
Forward log-odds (distinct)	**0.586**	***0.777***	**0.764**	**0.760**
Pair tri-gram baseline	0.604			

Table 4.12: *English-Chinese Pair HMM transliteration detection accuracy and MRR.*

Pair HMM algorithm	PHMM0	PHMM3	PHMM5	PHMM9
	accuracy			
Viterbi	0.639	0.651	0.671	0.733
Forward	0.733	0.755	**0.804**	0.807
Vit. log-odds (identical)	0.284	0.464	0.253	0.349
Vit. log-odds (distinct)	0.585	0.787	0.636	0.729
Forward log-odds (identical)	0.429	0.553	0.307	0.426
Forward log-odds (distinct)	**0.843**	***0.885***	0.793	**0.870**
Pair tri-gram baseline	0.813			
	MRR			
Viterbi	0.710	0.745	0.754	0.808
Forward	0.817	0.831	**0.866**	0.867
Vit. log-odds (identical)	0.396	0.576	0.345	0.458
Vit. log-odds (distinct)	0.683	0.853	0.724	0.805
Forward log-odds (identical)	0.549	0.665	0.414	0.538
Forward log-odds (distinct)	**0.899**	***0.923***	0.861	**0.911**

Table 4.13: *English-Hindi Pair HMM transliteration detection accuracy and MRR.*

Pair HMM algorithm	PHMM0	PHMM3	PHMM5	PHMM9
	accuracy			
Viterbi	0.593	0.602	0.780	0.758
Forward	0.636	0.693	0.816	0.841
Vit. log-odds (identical)	0.342	0.422	0.426	0.411
Vit. log-odds (distinct)	0.698	0.769	0.801	0.762
Forward log-odds (identical)	0.326	0.447	0.490	0.479
Forward log-odds (distinct)	**0.779**	**0.855**	***0.858***	**0.854**
Pair tri-gram baseline	0.763			
	MRR			
Viterbi	0.659	0.701	0.849	0.831
Forward	0.746	0.790	0.873	0.891
Vit. log-odds (identical)	0.460	0.535	0.546	0.526
Vit. log-odds (distinct)	0.782	0.836	0.862	0.831
Forward log-odds (identical)	0.448	0.574	0.608	0.603
Forward log-odds (distinct)	**0.858**	**0.904**	***0.905***	**0.903**
Pair tri-gram baseline	0.806			

Table 4.14: *English-Kannada Pair HMM transliteration detection accuracy and MRR.*

Pair HMM algorithm	PHMM0	PHMM3	PHMM5	PHMM9
	accuracy			
Viterbi	0.688	0.781	0.876	0.878
Forward	0.810	0.818	0.881	0.876
Vit. log-odds (identical)	0.365	0.740	0.823	0.828
Vit. log-odds (distinct)	0.633	0.866	0.878	0.884
Forward log-odds (identical)	0.472	0.731	0.836	0.841
Forward log-odds (distinct)	**0.864**	**0.885**	***0.888***	**0.887**
Pair tri-gram baseline	0.944			
	MRR			
Viterbi	0.738	0.821	0.884	0.885
Forward	0.843	0.846	0.888	0.885
Vit. log-odds (identical)	0.442	0.787	0.850	0.854
Forward log-odds (identical)	0.565	0.786	0.861	0.863
Forward log-odds (distinct)	**0.878**	**0.889**	***0.892***	**0.891**
Pair tri-gram baseline	0.956			

Table 4.15: *English-Russian Pair HMM transliteration detection accuracy and MRR.*

Pair HMM algorithm	PHMM0	PHMM3	PHMM5	PHMM9
	accuracy			
Viterbi	0.447	0.558	0.681	0.652
Forward	0.559	0.679	0.766	0.773
Vit. log-odds (identical)	0.227	0.347	0.227	0.190
Vit. log-odds (distinct)	0.649	0.726	0.709	0.660
Forward log-odds (identical)	0.201	0.383	0.281	0.223
Forward log-odds (distinct)	**0.702**	**0.824**	***0.827***	**0.789**
Pair tri-gram baseline	0.775			
	MRR			
Viterbi	0.529	0.670	0.771	0.741
Forward	0.684	0.780	0.843	0.848
Vit. log-odds (identical)	0.342	0.463	0.346	0.278
Vit. log-odds (distinct)	0.745	0.810	0.796	0.746
Forward log-odds (identical)	0.315	0.506	0.420	0.320
Forward log-odds (distinct)	**0.799**	***0.889***	**0.886**	**0.861**
Pair tri-gram baseline	0.824			

Table 4.16: *English-Tamil Pair HMM transliteration detection accuracy and MRR.*

Pair HMM algorithm	PHMM0	PHMM3	PHMM5	PHMM9
	accuracy			
Viterbi	0.390	0.348	0.468	0.474
Forward	0.490	0.458	0.518	0.520
Vit. log-odds (identical)	0.141	0.165	0.150	0.156
Vit. log-odds (distinct)	0.602	0.646	0.600	0.589
Forward log-odds (identical)	0.150	0.199	0.170	0.176
Forward log-odds (distinct)	**0.670**	***0.793***	**0.679**	**0.668**
Pair tri-gram baseline	0.590			
	MRR			
Viterbi	0.521	0.478	0.588	0.590
Forward	0.628	0.595	0.637	0.635
Vit. log-odds (identical)	0.250	0.280	0.255	0.261
Vit. log-odds (distinct)	0.709	0.748	0.698	0.693
Forward log-odds (identical)	0.261	0.325	0.282	0.285
Forward log-odds (distinct)	**0.772**	***0.865***	**0.774**	**0.765**
Pair tri-gram baseline	0.659			

Table 4.17: *English-Thai Pair HMM transliteration detection accuracy and MRR.*

Pair HMMs to result in better transliteration detection quality since Cyrillic and Latin alphabets that are used in Russian and English respectively are both mostly phonemic and character correspondences between the two should not be difficult to induce using Pair HMMs. However, as Table 4.15 shows, none of the Pair HMMs results in an accuracy or MRR higher than that for the baseline approach of using Pair trigram and higher order n-gram information for computing transliteration similarity (Table 4.10). In order to eliminate the chance that we could have used an inadequately trained model, we applied the Baum-Welch algorithm for each of the Pair HMM variants on the English-Russian datasets this time with random probability distributions for the initial parameters, but there was no change in transliteration detection accuracy and MRR of the Pair HMMs compared to the baseline approach. Probably, the unique underperformance of the Pair HMMs compared to the baseline approach in the results above can be attributed to the small English-Russian transliteration data we used for evaluation. Despite the result for the English-Russian language pair, the Pair HMMs achieve considerably higher transliteration detection accuracy and MRR for all the other language pairs in comparison to the performance of the baseline approach that uses pair tri-gram information.

4.6 Conclusion

In this chapter, we have evaluated several Pair HMM settings in an experimental transliteration detection task. In the first set of experiments where we evaluate the models in identifying transliterations on a geographic names corpora, results show that it is important to use models that accurately capture the properties of character representations in different writing systems. A comparison of the transliteration similarity scoring algorithms shows that it is important to factor in a measure for the random probability of relating two candidate name pairs. We see that the Forward log-odds algorithm consistently leads to the best transliteration detection accuracy and MRR. We also see that the use of a *log-odds* score always results in better transliteration detection quality across all model settings. The results also suggest the insensitivity of using the log-odds similarity score on changes in model parameterizations. For example, we see that the Pair HMM variant that uses only 3 transition parameters and a Forward log-odds algorithm leads to the best overall performance despite a poor performance by the other Pair HMM algorithms using this model. An investigation into changes in Pair HMM state transition parameters shows that is important to use transition parameters in the computation of transliteration similarity. However, the results also suggest that an increase in transition parameters does not guarantee an improvement in transliteration detection quality. Generally, results in this chapter suggest that the edit distance-based approach of Pair HMMs could be valuable in searching for transliteration pairs from bilingual text.

Chapter 5

Transduction-based DBN models for transliteration detection

5.1 Introduction

So far, we have adapted the approach of Pair Hidden Markov models (Pair HMMs) to compute transliteration similarity with promising results in an experimental transliteration detection task. In this chapter, we introduce the second edit distance-based Dynamic Bayesian Network (DBN) modeling approach that we adapt to compute transliteration similarity. This approach is founded on the representation of a memoryless stochastic transducer (Ristad and Yianilos 1997) as an edit distance-based DBN model (Filali and Bilmes 2005). We have referred to this approach as the 'transduction-based DBN approach'. Unlike the Pair HMM approach whose implementation is based on the classic Hidden Markov model (HMM) framework, the transduction-based DBN approach in this chapter is implemented using the probabilistic graphical modeling framework where templates are used to represent various types of DBN models. The use of DBN templates enables us to define several generalizations of DBN models with the possibility of overcoming restrictions that are fundamental to the framework for classic HMMs. For example, it is possible to relax HMM restrictions associated with the Markovian assumptions of conditional independence and the definition of an HMM as a function of a single independent variable. The approach enables us to use an arbitrary number of random variables

This chapter is an extended version of the following publications:
P. Nabende – Applying a Dynamic Bayesian Network framework to transliteration identification, *Proceedings of the* 7^{th} *International Conference on Language Resources and Evaluation Conference* (LREC 2010), pp. 244–251, May 2010, Valletta, Malta; and
P. Nabende – Comparison of applying Pair HMMs and DBN models in transliteration identification, *Proceedings of the* 20^{th} *Computational Linguistics in Netherlands meeting*, pp. 107–122, Feb 2010, Utrecht, The Netherlands.

and dependencies between the variables which we can associate with various factors that we hypothesize to have an effect on transliteration detection quality.

In the next section, we review the origins of the transduction-based DBN approach. Later, we describe different DBN model generalizations that we have adapted to compute transliteration similarity. We then present the results from applying some transduction-based DBN models in an experimental transliteration detection task using the same evaluation setup described in the previous chapter. This enables us to compare transliteration-detection results from using transduction-based DBN models to results from using Pair HMMs and the baseline approach of pair n-gram models.

5.2 Transduction-based DBN models

The transduction-based DBN approach (Filali and Bilmes 2005) is founded on the representation and implementation of a *memoryless stochastic transducer* (initially proposed by Ristad and Yianilos (1997)) as a DBN model for learning string edit distance. It is helpful to review the formulation of the *memoryless stochastic transducer* so as to understand the random variables and conditional probability distributions defined in the construction of the corresponding DBN model.

5.2.1 The memoryless stochastic transducer

Ristad and Yianilos (1997) model string edit distance as a memoryless stochastic transduction between two strings. The transductions involve any of the three edit operations of generating a substitution pair of symbols denoted by ⟨a,b⟩, a deletion pair $\langle a,\epsilon\rangle$, an insertion pair $\langle \epsilon, b\rangle$ or a termination. In order to compute the probability for a pair of strings, Ristad and Yianilos consider a string pair to be an equivalence class representative for all edit sequences whose yield is that string pair. The probability of a string pair is therefore computed as the sum of the probabilities of all edit sequences for that string pair.

Assuming that the source string is denoted by $s_1^m = s_1 s_2 ... s_m$ where the characters are associated with the source alphabet A_s, and the target string is denoted by $t_1^n = t_1 t_2 ... t_n$ based on a target alphabet A_t. The edit operations can be represented using a hidden random variable Z that takes in values from $(A_s \cup \epsilon \times A_t \cup \epsilon) \setminus (\epsilon, \epsilon)$. Following Ristad and Yianilos' consideration in the last paragraph, Z can be perceived as a random vector with two components ($Z^{(s)}$ and $Z^{(t)}$). The probability of a pair of strings can therefore be computed by marginalizing over the two components of the edit operation variable Z (Filali and Bilmes 2005):

$$P(s_1^m, t_1^n|\theta) = \sum_{z^{(s)}} \sum_{z^{(t)}} P(z_1^l, s_1^m, t_1^n|\theta) \tag{5.1}$$

where θ represents the parameters for the memoryless stochastic transducer and z_1^l is such that its yield denoted by $v(z_1^l) = \langle s_1^m, t_1^n \rangle$ and $\max(m,n) \leq l \leq m+n$.

In the memoryless stochastic transducer, there is no dependence between edit operations. Therefore $P(z_1^l, s_1^m, t_1^n|\theta)$ is simply the product of the probabilities of the individual edit operations, that is

$$P(z_1^l, s_1^m, t_1^n|\theta) = \prod_i P(z_i, s_1^m, t_1^n|\theta) \text{ for } 1 \leq i \leq l \text{ and } z_i = \langle z_i^{(s)}, z_i^{(t)} \rangle$$

The RY memoryless stochastic transducer is also context-independent in the sense that the edit operation random variable (z_i) does not have any local dependence on the source and target string characters (s_{a_i} and t_{b_i} respectively), but does have a *global context dependence* that ensures the generation of the string pair (s_m, t_n). Using $Q(z_i)$ to represent $P(z_i, s_1^m, t_1^n|\theta)$, this local context independence is depicted in the following expression (Filali and Bilmes 2005):

$$Q(z_i) \propto \begin{cases} f^{ins}(t_{b_i}) & \text{for } z_i^{(s)} = \epsilon; z_i^{(t)} = t_{b_i} \\ f^{del}(s_{a_i}) & \text{for } z_i^{(s)} = s_{a_i}; z_i^{(t)} = \epsilon \\ f^{sub}(s_{a_i}, t_{b_i}) & \text{for } (z_i^{(s)}, z_i^{(t)}) = (s_{a_i}, t_{b_i}) \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

where $\sum_z Q(z) = 1$; $a_i = \sum_{j=1}^{i-1} 1_{\{z_j^{(s)} \neq \epsilon\}}$ and $b_i = \sum_{j=1}^{i-1} 1_{\{z_j^{(t)} \neq \epsilon\}}$ are the respective indexes of the source and target string generated up to the i^{th} edit operation; and f^{ins}, f^{del}, and f^{sub} are the functions that map to [0,1] corresponding to the three edit operation types of insertion, deletion, and substitution. Equation 5.2 also enforces the consistency constraint that the pair of characters output by $(z_i^{(s)}, z_i^{(t)})$ is the same as the actual pair of characters (s_{a_i}, t_{b_i}) that needs to be generated at the i^{th} transduction so that the total yield, $v(z_1^l)$ is equal to the string pair (s_1^m, t_1^n).

5.2.2 Representing the RY transducer as a DBN

Filali and Bilmes (2005) use the graphical models framework to represent the joint probability distribution of the Ristad-Yianilos (RY) stochastic transducer model with its related consistency constraints as a DBN model. Random variables are defined for the DBN model to correspond to the objects that contribute to the computation of edit distance based on the memoryless stochastic transducer model. From the description in the previous section, the main objects of interest include: the edit operation variable (Z_i); source and target character variables; variables that capture the position of the characters in the source and target strings; and consistency variables that check the yield of the edit operation variable against the actual pair of characters at a given position. The dependencies between the random variables follow

naturally. For example, a dependency is defined between a string position variable and the character variable. In this case, the idea is that knowledge about the position in a string leads to knowledge about the character at that position. For consistency checking, the consistency variables are defined to depend on the character variables and the edit operation variable. To complete the specification of the DBN model, we need to define the initial Bayesian network ($\mathcal{B}_0$), and a 2-frame Temporal Bayes Net (2-TBN) ($\mathcal{B}_{\rightarrow}$). We use the same graphical modeling approach in Filali and Bilmes (2005) where the main modeling terms are associated with concepts in Automatic Speech Recognition (ASR).

In the ASR-based graphical modeling approach, a *frame*[1] is used to represent a set of random variables and their attributes at a given time. The term *Prologue frame* is used to refer to the initial Bayesian network $\mathcal{B}_0$, and the term *Chunk frame* is used to refer to the Bayesian network that is to be *unrolled* as defined for a 2-TBN. An *Epilogue frame* refers to a Bayesian network that models end conditions. Using this ASR-based notation, the initial, chunk, and epilogue Bayesian networks that model the memoryless stochastic transducer are as shown in Figure 5.1. Since the networks are designed to model the properties of the memoryless stochastic transducer, the DBN template defined by the three networks in Figure 5.1 is also referred to as *memoryless and context-independent* (MCI). In the following paragraph, we describe the variables and dependencies used to specify the MCI DBN template in Figure 5.1.

The nodes in Figure 5.1 represent variables while the directed edges represent 'informational relationships' between the variables. As defined in chapter 3, the directed edges are used to suggest the influence of one variable on another. In Figure 5.1 Z as defined above is the edit operation variable and is defined for the DBN model to take values from a distribution whose cardinality is equal to the product of the size of the source ($n_{A_s \cup \epsilon}$) and target ($n_{A_t \cup \epsilon}$) vocabularies excluding (ϵ, ϵ). *sp* and *tp* are variables used to capture the current position in the source and target strings respectively. *sp* has a cardinality (m) equal to the length of the source string while *tp* has a cardinality (n) equal to the length of the target string. The s and t variables represent the current character in the source and target strings respectively. The cardinality of s is equal to the size of the source language alphabet including the empty string ($A_s \cup \epsilon$) while that for t is equal to the size of the target language alphabet ($A_t \cup \epsilon$). The *sc* and *tc* nodes enforce the consistency constraint[2] implied

[1] The term "frame" in the context of ASR, refers to contagious, small regions of a speech signal which aid in the identification of phonemes

[2] Following equation 5.2, the *sc* and *tc* nodes have a fixed observed value 1 and the only configuration of their parents is such that the source component of the edit operation variable Z is s or an empty symbol for *sc* and the target component of the edit operation Z is t or an empty symbol for *tc* and that Z does not generate empty source and target symbols at the same time (Filali and Bilmes 2005).

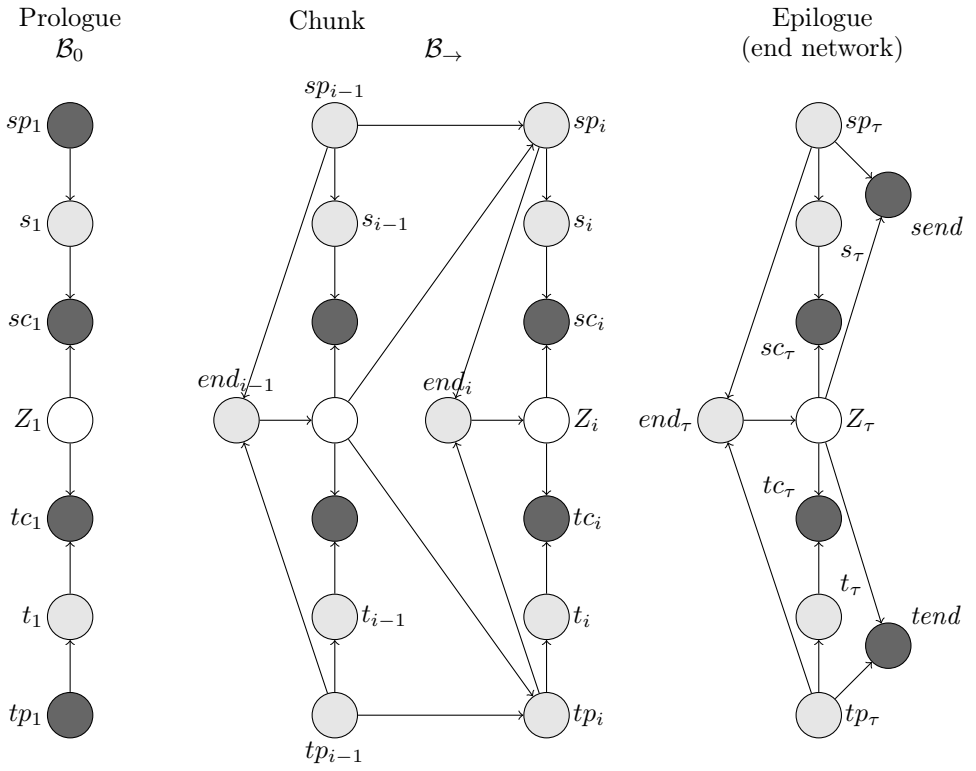

Figure 5.1: *Graphical representation for the MCI DBN template. Following the common convention for representing graphical models, shaded nodes represent observed variables, unshaded nodes represent hidden nodes, and nodes with dots represent deterministic hidden variables (Adapted from Filali and Bilmes, 2005).*

by equation 5.2. The *end* node is introduced for the DBN model as a 'switching' parent of Z and it is used to indicate when we are past the end of both the source and target strings, that is, when $sp > m$ and $tp > n$. The *send* and *tend* nodes represent variables that ensure that we are past the end of the source and target strings respectively.

Decision trees are used to implement deterministic conditional probability tables for most of the dependencies in Figure 5.1. Specifically, decision trees are used for each of the following variable dependencies: $sp_i \rightarrow s_i$ and $tp_i \rightarrow t$ (for the relationship between position and character variables of the source and target strings respectively); $s_i \rightarrow sc_i \leftarrow Z_i$ and $Z_i \rightarrow tc_i \leftarrow t_i$ (for the relationship between the consistency

variables and their respective parents); $sp_{i-1} \rightarrow sp_i \leftarrow Z_{i-1}$ and $tp_{i-1} \rightarrow tp_i \leftarrow Z_{i-1}$ (for transition on the source and target side respectively); $spos_i \rightarrow end_i \leftarrow tpos_i$ (for determining whether we are past the end of the edit sequence); $end_i \rightarrow Z_i$ (for determining a value that Z_i should have depending on the value (0 or 1) of end_i, that is end_i is a switching parent of Z_i, when $end_i = 0$, the conditional probability distribution of Z_i is as described above with a cardinality of $n_{A_s \cup \epsilon} \times n_{A_t \cup \epsilon}$, and when $end_i = 1$, Z_i takes with a probability 1, a fixed value outside the range of edit operations but consistent with s_i and t_i); $sp_i \rightarrow send_i \leftarrow Z_i$ and $tp_i \rightarrow tend_i \leftarrow Z_i$ (for determining whether we are at or past the end of the source and target strings respectively. This is not the same as for the end_i variable where we only check whether we are past the end of the edit sequence. So if $sp_i > m$ (the length of the source string), then $send_i$ is observed to be 1 with a probability 1; else if $send_i < m$, then $P(send_i = 1) = 0$ and the whole edit sequence is considered to to have zero probability; else if $sp_i = m$, then $send_i$ will have a probability greater than 1 only if the Z_i is an insertion). For the Z_i variable, a Dense Probability Mass Function (DPMF) is used to implement a Dense Conditional Probability Table (DCPT) which is used in the generation of source and target symbols in each frame.

5.2.3 DBN templates for modeling transliteration similarity

In order to propose additional edit distance based DBN templates using Filali and Bilmes' (2005) approach, we have to use the MCI DBN template as the baseline template. Our starting point would be to represent the classic HMMs which are the simplest type of DBN models. In this section, we discuss the requirements for representing Pair HMMs as edit distance-based DBN models based on the MCI DBN template.

First of all we compare Pair HMMs representation to transduction-based DBN modeling representation based on a comparison made by Zweig and Russell (1999) between DBNs and the classic HMMs for automatic speech recognition. We use an alignment example to help clarify the differences in representation. Consider an arbitrary one-to-one character alignment as illustrated in Table 5.1 between the Russian name 'Анатольевич' and its Dutch representation 'Anatoljevitsj'. The

А	н	а	т	о	л	ь	е	в	и		ч	
A	n	a	t	o	l	j	e	v	i	t	s	j
M	M	M	M	M	M	M	M	M	M	I	M	I

Table 5.1: *An arbitrary alignment between a Russian named "Анатольевич" and its Dutch representation "Anatoljevitsj".*

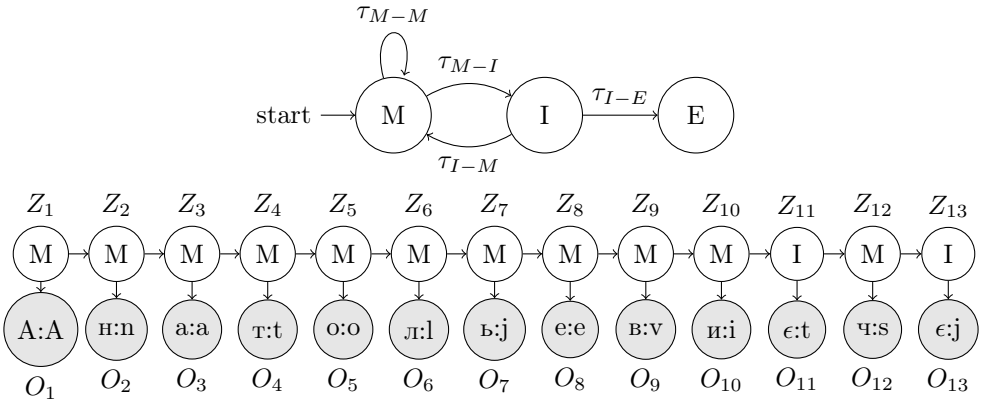

Figure 5.2: *A Pair HMM (top) and a conceptual DBN (bottom) representation for the alignment in Table 5.1. Shaded nodes in the DBN representation indicate 'observed' variables and unshaded ones indicate 'hidden' variables. ϵ denotes the 'empty' symbol.*

upper part of Figure 5.2 is a finite state representation of a Pair HMM for the alignment whereas the lower diagram is a corresponding conceptual DBN representation. A comparison of the two diagrams in Figure 5.2 shows at least two differences. One difference is that the nodes and arcs in the Pair HMM representation refer to states and transitions respectively, whereas in the DBN representation, nodes represent variables and arcs specify conditional dependencies / independencies between the variables. The other difference is that the labeling of the hidden variable (Z_i) shows that the DBN representation is explicit about time.

Although the DBN representation in Figure 5.2 seems to capture the Pair HMM representation, it does not yet represent the states of the Pair HMM at each time frame. If we follow the graphical modeling approach in the previous section, we need to introduce position variables in a manner similar to that as described for the MCI DBN model. The position variables will enable the representation of a Pair HMM at particular time frames. Figure 5.3 shows the DBN representation of the Pair HMM above with the position variables added; if we have a substitution operation, the source and target position variables are incremented; if we have an insertion, the source position variable will retain its previous value while the target position variable is incremented; and if we have a deletion, the target position variable retains its previous value while the source position variable is incremented. We can specify the Z variable to model the three Pair HMM emission states by taking values from a distribution with a cardinality of $n_{A_s \cup \epsilon} \times n_{A_t \cup \epsilon} - 1$. The Pair HMM transition probabilities can be encoded in the transition variables (that is between the edit

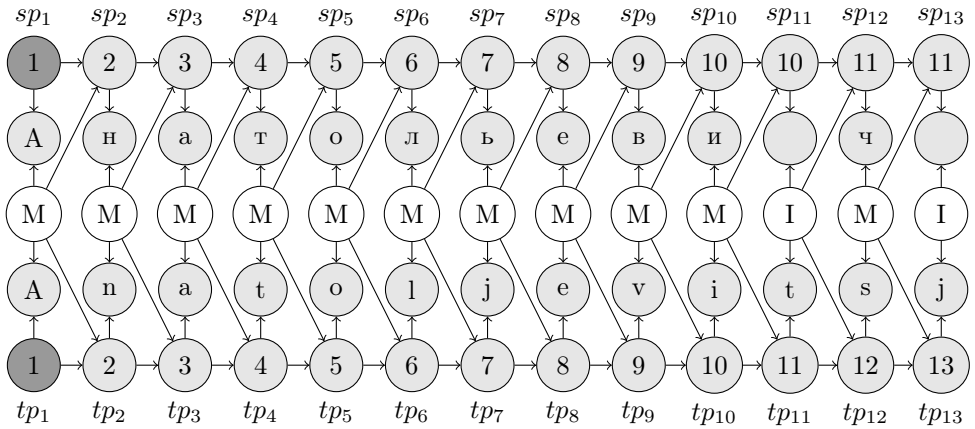

Figure 5.3: *A DBN representation of a Pair HMM with an illustrative assignment of values to the variables. Light gray nodes are used here to illustrate partially deterministic variables. Dark gray nodes represent deterministic variables while unshaded nodes represent hidden variables.*

operation variable Z_{i-1} and the position variables (sp_i and tp_i)). We could then use decision trees to implement the CPTs of the different dependencies in a similar way as was described for the MCI DBN model.

Based on the MCI DBN template, we adapt three other DBN model templates that were initially introduced by Filali and Bilmes (2005) for computing transliteration similarity. The three DBN model templates represent different types of dependencies on the edit operation variables. The three types of dependencies include: edit operation memory dependencies; source and / or target character context dependencies; and edit operation length dependencies. In the following, we briefly point out the main properties of these other DBN models.

a) Context-independent memory DBN template

Figure 5.4 shows the initial Bayesian network and the 2-TBN for the context independent memory (MEM) DBN template. Here, modeling *memory*, means having the current edit operation variable Z_i use information from the previous edit operation variable Z_{i-1}. In Figure 5.4, a variable H is introduced to model various dependencies between Z_{i-1} and Z_i. Generally, H can be stochastic or deterministic, and the amount of information that it summarizes from one frame to another is determined by its cardinality (Filali and Bilmes 2005). In this chapter, we investigate the deterministic implementation of H, where H enables the modeling of the conditional

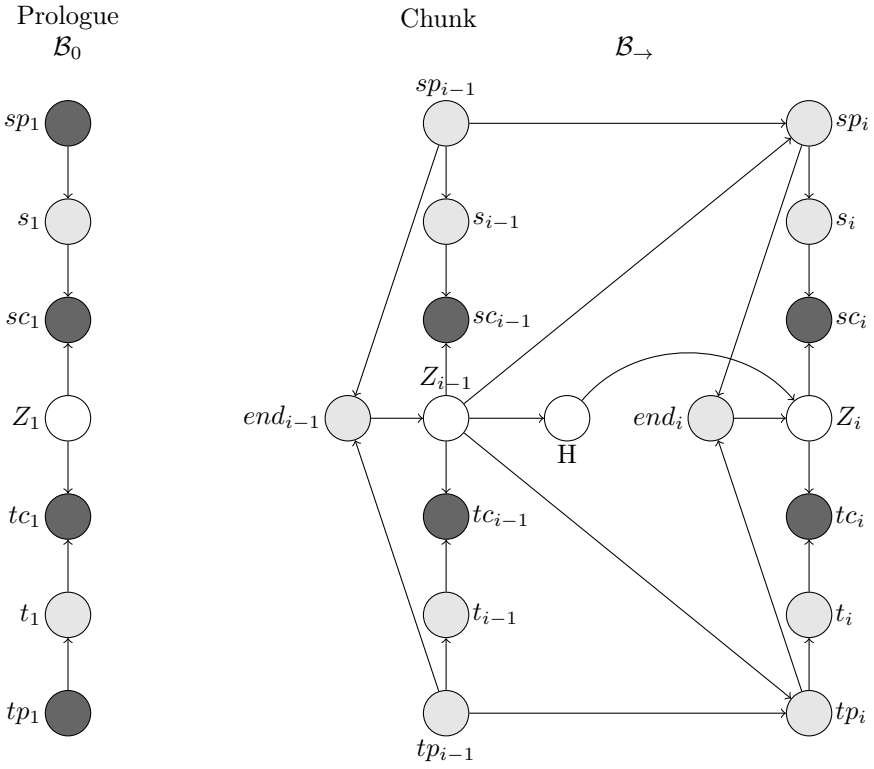

Figure 5.4: *Graphical representation for a context-independent memory DBN template. (Adapted from Filali and Bilmes, 2005).*

probability distribution $P(Z_i|Z_{i-1})$. Apart from this type of dependency, Z_i can also depend on the type of edit operation in the previous frame.

b) Context-dependent DBN template

Figure 5.5 shows the initial Bayesian network and the 2-TBN for a context-dependent DBN template. Context-dependence, here, means adding a dependence of the edit operation variable on the source and / or target string characters. For example, as shown in Figure 5.5, we model context-dependence of the edit operation on the source string characters by adding edges from s_i, $sprev_i$ to Z_i. We also see from Figure 5.5 that the source consistency variable *sc* is not used because the consistency constraint is defined through the conditional probability table of Z given its parents. Generally,

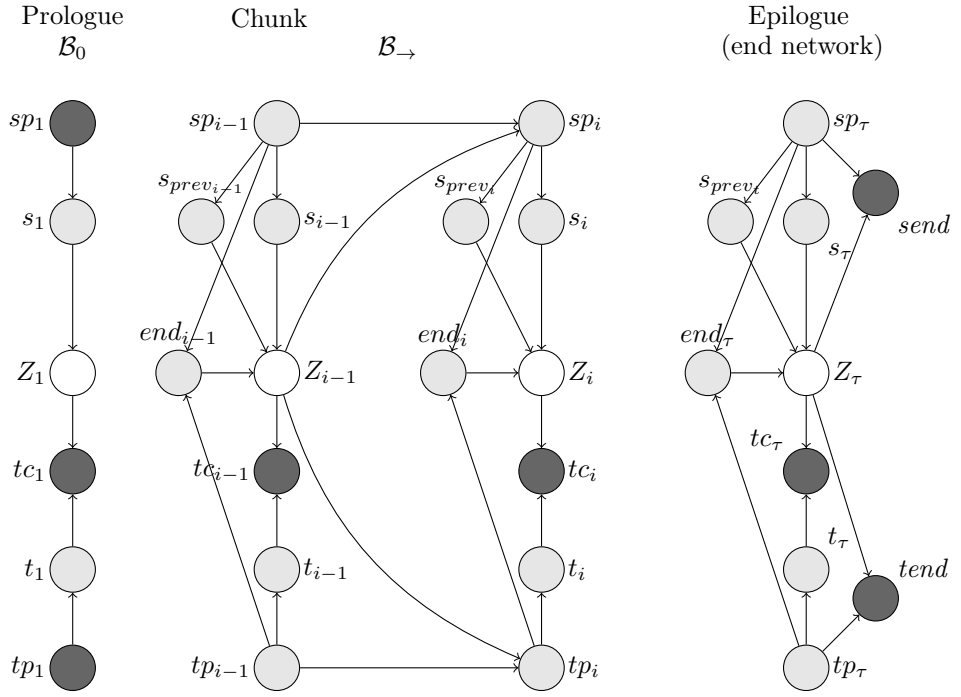

Figure 5.5: *Graphical representation for a context-dependent DBN template.*

we can model context-dependence to range from the case where we include only the dependence on the current character to the case where we consider all characters in the source (and / or target) string. In this chapter, we evaluate two main cases: one case is as shown in Figure 5.5; in the other case, we use only one depedence of the edit random variable on the current character in the source string. Based on these two cases, we evaluate context-dependence for two additional settings where we simply change the datasets so that the source becomes target and the target becomes source.

c) Length DBN template

Figure 5.6 shows the initial Bayesian network and the 2-TBN for a context-dependent *length* DBN template. To enable an investigation on the effect on transliteration detection quality from using a transduction-based DBN model where information about the length of the edit sequence is factored into the computation for transliteration similarity, we use additional variables which model the logic necessary for simulating

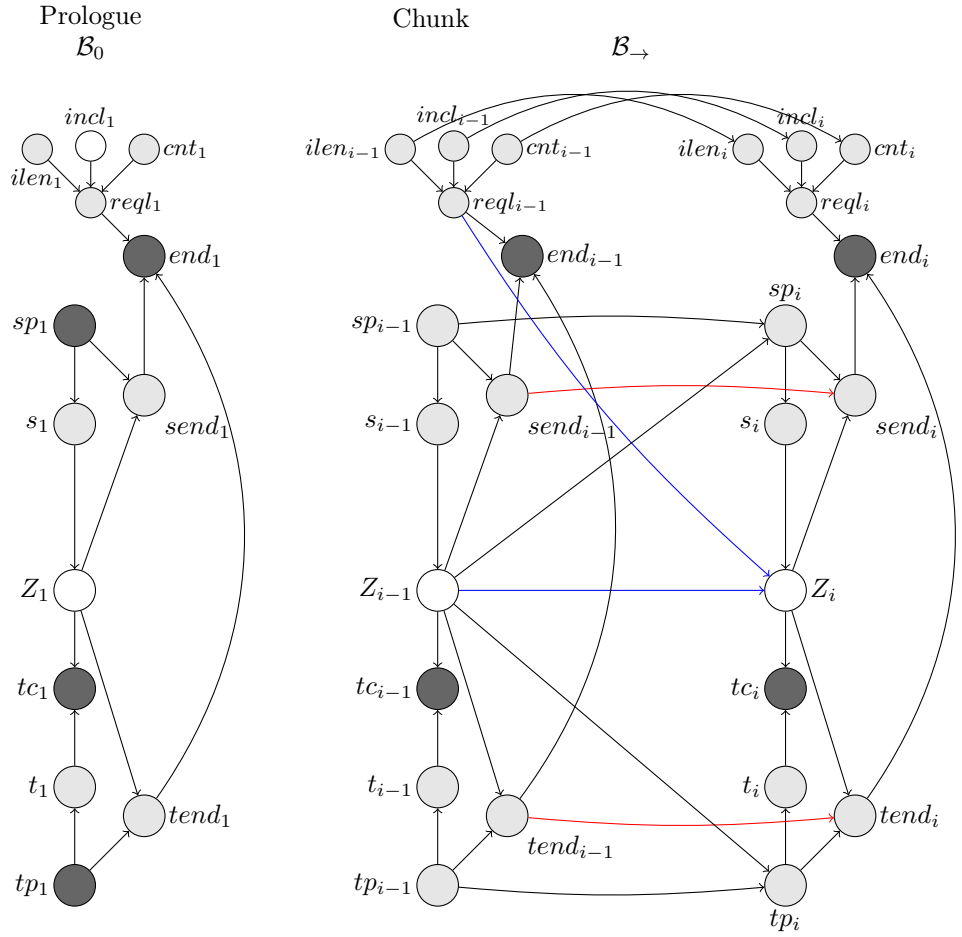

Figure 5.6: *Graphical representation for a context-dependent length DBN template.*

variable length-unrolling of the chunk frame. The additional variables in Figure 5.6 are defined as follows. *incl* is a stochastic hidden random variable whose value added to that of the variable *inilen* determines the number of allowed edit operations. The variable *cnt* is used to determine the index of the current edit operation and is used to trigger the random variable *reql* when the required sequence length is reached. The variable *end* is explained if we reach the end of one string after having reached the end of the other string in a previous frame. Apart from the template in Figure

5.6, we also investigate the effect of length for a context-independent DBN model by simply adding similar edges introduced by the additional variables in this section to the MCI DBN template in Figure 5.1.

d) Transliteration data requirements

The application of the DBN model templates above in the specification of DBN models to compute transliteration similarity requires the representation of transliteration data in a form suitable for running DBN inference algorithms. Most of the requirements which we discussed for the application of the Pair HMMs to compute transliteration similarity apply for the DBN templates introduced in this chapter. We follow the same segmentation approach as discussed in Chapter 4. That is we tokenize source and target strings per character. We also assume a monotonic ordering of the strings, and a one-to-one correspondence between tokens. The limitations described for the Pair HMMs based on these assumptions also apply to the DBN templates. In this chapter, we specify the DBN templates so that the resulting DBN models are based on the representation of separate source and target alphabets corresponding the respective writing systems. However, we do not investigate the effect of using separate alphabets against the use of one alphabet on the quality of transliteration similarity estimates as we did in Chapter 4.

5.2.4 Inference

We define the DBN templates for computing transliteration similarity using the Graphical models Toolkit (GMTK) (Bilmes and Zweig 2002). Our interest is in two main inference tasks when applying the edit distance-based DBN templates: parameter estimation given training transliteration data, and using the trained DBN models to compute transliteration similarity. In both cases, GMTK uses the *frontier algorithm* which we describe in the following.

a) Frontier algorithm

The frontier algorithm (Zweig 1996) uses a Forwards-Backwards procedure which updates the joint distribution over a set of hidden nodes without needing to create and manipulate a huge transition matrix as is the case in the Forwards-Backwards algorithm for the classic HMMs. The forwards-backwards procedure in both cases assumes that a hidden node (Z_t) or set of hidden nodes ($Z_t^{(1:D)}$) at a given time t d-separates the past from the future. For the DBNs, the Frontier algorithm uses a *Markov blanket* over the hidden nodes which it "sweeps" across the DBN first in the forwards direction (while computing the forward variable α) and then backwards

(for computing the backward variable β) (Murphy 2002). Our review of the Frontier algorithm follows from a simplified presentation of the same by Murphy (2002). We also use the same notation in (Zweig 1996).

Forwards pass

The nodes in the Markov blanket are called the "Frontier set" and the set is denoted by F; the nodes to the left and right of F are denoted by L and R. At every step of the frontier algorithm, F should d-separate L and R.

Let h_F refer to the hidden nodes in F, e_F to the evidence nodes in F, e_L to the evidence nodes in L, and e_R to the evidence nodes in R. In the forward pass, the probability of the nodes in F is expressed as $P(F) \stackrel{\text{def}}{=} P(h_F, e_F, e_L)$, and is computed recursively as follows.

We can add a node N to the frontier set (that is move it from R to F) when all its parents are already in F:

$$P(e_L, e_F, h_F, N) = P(e_L, e_F, h_F)p(N|e_F, h_F) \tag{5.3}$$

since N is conditionally dependent on only e_F and h_F but not on e_L. Equation 5.3 means that adding a node consists of multiplying its conditional probability distribution (CPD) into the frontier.

We can remove a node N from F to L when all its children are in F. If N is hidden, then $e_{L\cup\{N\}} = e_L$ and $e_{F\setminus\{N\}} = e_F$.

$$\begin{aligned} P(e_{L\cup\{N\}}, e_{F\setminus\{N\}}, h_{F\setminus\{N\}}) &= P(e_L, e_F, h_{F\setminus\{N\}}) \\ &= \sum_N P(e_L, e_F, N, h_{F\setminus\{N\}}) \\ &= \sum_N P(e_L, e_F, h_F) \end{aligned}$$

which means that removing a node consists of marginalizing it out. The same applies to the case where N is observed:

$$\begin{aligned} P(e_{L\cup\{N\}}, e_{F\setminus\{N\}}, h_{F\setminus\{N\}}) &= P(e_{L\cup\{N\}}, e_{F\setminus\{N\}}, h_F) \\ &= P(e_L, e_N, e_{F\setminus\{N\}}, h_F) \\ &= P(e_L, e_F, h_F) \end{aligned}$$

Backwards Pass

In the backwards pass, $P(F) \stackrel{\text{def}}{=} P(e_R|h_F, e_F)$. The frontier is advanced from frame $t+1$ to frame t by adding and removing nodes in the opposite order that is used in the forwards pass. Adding a node in this case means moving it from L to F, and removing a node means moving it from F to R.

When a node N is added to F, we compute $P(e_R|e_F, h_F, N)$. In this case, since N's children are in F, which "shield" N from e_R; $P(e_R|e_F, h_F, N) = P(e_R, |e_F, h_F)$. When a node N is removed from F, and added to R, we compute $P(e_{R\cup\{N\}}|e_{F\setminus\{N\}}, h_{F\setminus\{N\}})$ from $P(e_R|e_F, h_F)$. If N is hidden, then $e_{R\cup\{N\}} = e_R$, and $e_{F\setminus\{N\}} = e_F$. Therefore

$$\begin{aligned} P(e_{R\cup\{N\}}|e_{F\setminus\{N\}}, h_{F\setminus\{N\}}) &= P(e_R|e_F, h_{F\setminus\{N\}}) \\ &= \sum_N P(N, e_R|e_F, h_{F\setminus\{N\}}) \\ &= \sum_N P(N|e_F, h_{F\setminus\{N\}}) P(e_R|N, e_F, h_{F\setminus\{N\}}) \\ &= \sum_N P(N|e_F, h_{F\setminus\{N\}}) P(e_R|e_F, h_F) \end{aligned}$$

This means that to remove a node N, we multiply in N's CPD.

The same procedure applies to the case where N is observed only that there is no need to marginalize out N since it has only one possible value.

$$\begin{aligned} P(e_{R\cup\{N\}}|e_{F\setminus\{N\}}, h_{F\setminus\{N\}}) &= P(e_{R\cup\{N\}}|e_{F\setminus\{N\},h_F}) \\ &= P(e_N, e_R|e_{F\setminus\{N\}}, h_F) \\ &= P(e_N|e_{F\setminus\{N\}}, h_F) P(e_R|e_N, e_{F\setminus\{N\}}, h_F) \end{aligned}$$

b) Generalized Expectation Maximization

In the transliteration detection experiments, we use GMTK's implementation of the generalized expectation maximization (GEM) algorithm to estimate parameters for each of the transduction-based DBN models. The reader is referred to Chapter 3 for a detailed explanation of Expectation Maximization (EM) and the case for GEM. The parameters for the DBN models constitute conditional probability tables that encode the relationships between variables in a frame and between two adjacent frames, and values for decision trees that are used to represent deterministic mappings. Specifically, there are two types of parameters (Bilmes 2002): numerical parameters which in our case include dense and sparse CPTs and for which an EM algorithm is used; and non-numerical parameters, which in our case constitute decision tree deterministic values that do not require the use of an EM algorithm.

5.3 Experiments (NEWS 2009 and 2010 shared task data)

5.3.1 Data

To ensure a comparison of the edit distance-based DBN models introduced in this chapter and the Pair HMMs in a transliteration detection task, we use transliteration data from the NEWS 2009 and 2010 shared tasks on transliteration generation. We have already described the transliteration data for seven language pairs (English-Bengali, English-Chinese, English-Hindi, English-Kannada, English-Russian, English-Tamil, and English-Thai) in Chapter 4, section 4.5.1.

5.3.2 Evaluation setup and results

We use the same evaluation setup described in Chapter 4, section 4.5.2 associated with the second set of experiments where we use transliteration data from the NEWS 2009 and NEWS 2010 shared tasks on transliteration generation. For the results reported in this section, we trained the transduction-based DBN models on the same training data (for each language pair) that we used for training the Pair HMMs and pair n-gram models. In estimating the parameters for each DBN model, we applied the generalized EM algorithm with a maximum specification of three iterations. During pre-runs, we observed this number of iterations to be optimal in avoiding the overfitting of the models. We then applied a scoring algorithm implemented in the GMTK toolkit to estimate transliteration similarity using the parameters of the trained DBN models. We also evaluate the transduction-based DBN models using *transliteration detection accuracy* and *Mean Reciprocal Rank* (MRR) as defined in Chapter 4.

Table 5.2 shows the transliteration detection accuracy and MRR results for the transduction-based DBN models as compared to the best performing Pair HMMs (Phmm) and the best baseline results for the Pair tri-gram method on the same transliteration data for the seven language pairs. In Table 5.2: MCI refers to the *memoryless and context-independent* DBN model (Figure 5.1), MEM refers to the *memory-dependent* DBN model (Figure 5.4), CONs1 and CONs2 represent context-dependent DBN models where the dependence of the edit operation on the characters in the source string is factored into the computation for transliteration similarity, and CONt1 and CONt2 represent context-dependent DBN models where the dependence of the edit operation on the target string character(s) is used. For CONs1, the edit operation depends on the current character in a source string whereas for CONs2, the edit operation depends on both the current and previous character in the source

Models	En-Be	En-Ch	En-Hi	En-Ka	En-Ru	En-Ta	En-Th
	accuracy						
MCI	0.868	0.299	0.747	0.718	0.978	0.646	0.351
MEM	0.894	0.487	0.717	0.576	0.888	0.717	0.489
CONs1	**0.960**	0.699	0.855	0.842	**0.983**	0.828	0.736
CONs2	0.957	0.796	0.857	0.845	0.974	0.853	0.789
CONt1	0.954	0.676	0.840	0.835	**0.983**	0.840	0.762
CONt2	**0.960**	**0.820**	0.859	0.856	0.976	**0.862**	**0.848**
LENs1	0.956	0.713	0.831	0.767	0.982	0.839	0.733
LENt1	0.954	0.698	0.847	0.806	0.976	0.797	0.700
best PHMM	0.932	0.684	**0.885**	**0.858**	0.888	0.827	0.793
pair tri-gram	0.804	0.524	0.813	0.763	0.944	0.775	0.623
	MRR						
MCI	0.910	0.423	0.827	0.806	0.985	0.751	0.479
MEM	0.914	0.623	0.774	0.634	0.921	0.809	0.612
CONs1	**0.975**	0.791	0.908	0.897	0.986	0.896	0.833
CONs2	0.972	0.869	0.908	0.901	0.979	0.913	0.868
CONt1	0.972	0.775	0.897	0.895	**0.987**	0.905	0.849
CONt2	**0.975**	**0.887**	0.908	**0.908**	0.981	**0.920**	**0.913**
LENs1	0.971	0.807	0.888	0.839	0.985	0.903	0.828
LENt1	0.970	0.786	0.901	0.871	0.981	0.867	0.791
best PHMM	0.950	0.777	**0.923**	0.905	0.892	0.886	0.865
pair tri-gram	0.844	0.604	0.850	0.806	0.956	0.824	0.685

Table 5.2: *Transliteration detection accuracy and MRR for different transduction-based DBN models against best Pair HMM results (best PHMM) and baseline approach of using Pair tri-gram information.*

string (Figure 5.5). CONt1 and CONt2 correspond respectively to CONs1 and CONs2, but with t referring to the target character(s). For results in this section, English was used as the target language and each of the other languages as the source language.

Table 5.2 shows that transduction-based DBN models improve transliteration detection accuracy and MRR for 5/7 language pairs (English-Bengali, English-Chinese, English-Russian, English-Tamil, and En-Thai). The transduction-based DBN models also post a performance comparable to that of the Pair HMMs on English-Kannada transliteration data. Unlike the case for the Pair HMMs, the transduction-based DBN models outperform the baseline approach of using pair n-gram information for

all language pairs including the case for the English-Russian data where Pair HMMs were outperformed by the baseline approach. The *memory-dependent* DBN model results in the lowest accuracy and MRR compared to all the other transduction-based DBN models; this result is unexpected in comparison to the MCI DBN model where no edit operation dependencies are used nor character context dependencies. The relatively better results from the context-dependent DBN models underline the necessity to represent character context dependencies for the edit operation random variable.

According to Table 5.2 results, the performance of the transduction-based DBN models in detecting transliterations is similar to that reported for the pronunciation classification task by Filali and Bilmes (2005) where context-dependent DBN models achieved the lowest word error rate. However, since we evaluate the transduction-based DBN models for more than one language pair in the task of detecting transliterations, we see some variations in transliteration detection accuracy. For the transliteration detection task, CONs1 and CONt2 models perform better than the the other context-dependent DBN models on the English-Bengali dataset while CONs1 performs best on the English-Russian dataset, and CONt2 performs best on the remaining datasets. In the pronunciation classification task (Filali and Bilmes 2005), CONs2 had the best result from their set of context-dependent DBN models.

Table 5.2 results also suggest that the language where we model context can have a considerable effect on transliteration detection accuracy. If we compare the results for CONs1 and CONt1, and those for CONs2 and CONt2, we see that modeling context on the target language side improves transliteration detection accuracy over the case where context is modeled on the source language side for almost all language pairs.

5.3.3 Error Analysis

The transliteration detection results in Table 5.2 clearly show that we still need to improve transliteration detection quality for most of the language pairs. An analysis of sample results for the same input could be helpful in finding ways of achieving higher transliteration detection quality when applying edit distance-based DBN models to compute transliteration similarity. Table 5.3 shows a sample of the returned results in the detection of Thai target transliterations given an English source word and where at least one of the context-dependent models did not return the correct candidate at *first rank*. From the sample results in the table, we see that the context-dependent DBN models result in different suggestions at the first rank. We also see that the reference named entity (NE) is returned at different ranks. The results in the table also suggest that if we had a way to combine the models so that system used only

Src. NE	Ref. NE	CONs1	CONs2	CONt1	CONt2
holeman	โฮล์เม่น	ฮัลต์เม่น hultman(3)	โฮลส์เมิน holzman(2)	(1)	โฮลส์เมิน holzman(4)
rickett	รีคิด	ริกโค reco(8)	ริกส์ riggs(13)	ไรกิ reiki(11)	ริชาร์ด richard(2)
dare	แดร์	ดาร์ก dark(6)	ดาร์ต dart(8)	เอเดอร์ ader(4)	ดาร์ก dark(3)
deibel	ไดเบิล	เอเบิล abell(4)	เอเบิล abell(10)	(1)	(1)
guide	ไกด์	กวิโด้ guido(22)	กวิโด้ guido(5)	เกลด์ geld(6)	เกลด์ geld(2)
wright	ไรต์	แวริ่ง wareing(7)	วาไรตี variety(2)	ฮาร์ต hart(25)	(1)
mier	ไมเออร์	เมเออร์ maier(2)	(1)	มัวร์ moore(4)	เมเออร์ maier(2)
mynn	มินน์	มีโนน(r2) minoan(2)	แยนต์(r3) yant(3)	มินน์(r2) minn(2)	มินน์(r4) minn(4)
harrod	แฮร์เริด	ฮาโรลด์(r2) harold(2)	ฮาร์วูด(r2) harwood(2)	แชร์เริด(r2) sherrod(2)	(1)
bowerman	โบเออร์เม่น	โอเวอร์เมิน overman(2)	โอเวอร์เมิน overman(2)	(1)	(1)
blacka	แบค่า	แบล็กเคอร์ blacker(23)	บอลติก baltic(7)	ไบก์ bike(11)	บังกา banka(2)
pettus	เพเติส	ฟีตัส foetus(2)	(1)	(1)	(1)
exmouth	เอ็กส์เมิท	สมูต smoot(14)	(1)	(1)	(1)
hostess	โฮสเตส	คอสต์ cost(3)	(1)	เชสเติร์ส chesters(2)	(1)
site	ไซต์	ซิตี city(4)	ซิตี city(2)	คอสต์ cost(4)	(1)

Table 5.3: *Examples of where at least one context-dependent DBN model failed to detect the correct transliteration at the first rank between English and Thai. The numbers in parentheses indicate the rank for the correct transliteration.*

the best result, we would improve transliteration detection accuracy in comparison to using the DBN models 'individually'. However, it is difficult to model this decision making process since we do not know which DBN model's similarity estimate will result in the best suggestion of a true transliteration given a source word and vice versa. But we can still explore other simple schemes of combining the models to determine whether there can be improvements in transliteration detection accuracy. In the next subsection we evaluate the effect of different combinations of context-dependent DBN models on transliteration detection accuracy and MRR.

5.3.4 Computing transliteration similarity based on ensembles of DBN models

The ensemble framework that we have used in combining context-dependent DBN models to compute transliteration similarity is quite straightforward. First of all, we estimate the parameters of each context-dependent DBN model separately. We then apply the trained models separately to compute transliteration similarity estimates for the candidate string pairs. Finally, we combine the individual model similarity estimates by adding them for each candidate string pair and getting an average based on the total number of models.

Table 5.4 shows the transliteration detection results where we have used the simple ensemble-based approach described in the last paragraph. Only context-dependent DBN models are combined since they achieved better transliteration detection accuracy and MRR compared to other transduction-based DBN models. In Table 5.4, CONs1, CONs2, CONt1, CONt2 are as described above whereas ALLCONst represents the combination of these four context-dependent DBN models. The values in **bold** format in Table 5.4 indicate the best overall result which includes a comparison with results in Table 5.2 where the edit distance-based DBN models were applied 'individually'.

Table 5.4 results suggest that using the simple scheme of summing transliteration similarity estimates from different DBN models and finding an average always leads to a fair improvement in transliteration detection accuracy and MRR (although slightly for some combinations) over the 'individual' application of the edit distance-based DBN models. Table 5.4 results also suggest that combining context-dependent models where character context is modeled in at least one DBN model on the source language side and in at least one DBN model on the target language side leads to better transliteration detection quality in comparison to the case of 'individual' application of a context-dependent DBN model where character context is modeled on only the source or target language side. This is also the case in (Kondrak and Sherif 2006) where an average context dependent DBN model and Pair HMMs outperform

Combination	En-Be	En-Ch	En-Hi	En-Ka	En-Ru	En-Ta	En-Th
	accuracy						
CONs1+CONs2	0.967	0.791	0.864	0.856	0.980	0.859	0.811
CONs1+CONt1	0.971	0.800	0.865	0.861	**0.987**	0.864	0.834
CONs1+CONt2	0.971	0.864	0.868	0.866	0.984	0.872	0.865
CONs2+CONt1	0.969	0.846	0.865	0.863	0.984	0.971	0.856
CONs2+CONt2	0.970	**0.883**	0.870	0.867	0.984	0.874	**0.879**
CONt1+CONt2	0.965	0.811	0.863	0.854	0.984	0.865	0.845
ALLCONst	**0.972**	0.872	0.870	**0.868**	0.984	**0.877**	0.875
	MRR						
CONs1+CONs2	0.979	0.866	0.912	0.908	0.984	0.917	0.886
CONs1+CONt1	**0.983**	0.871	0.912	0.912	**0.989**	0.921	0.902
CONs1+CONt2	0.982	0.918	0.914	0.914	0.987	0.926	0.924
CONs2+CONt1	0.980	0.905	0.914	0.913	0.986	0.925	0.918
CONs2+CONt2	0.980	**0.931**	0.916	0.915	0.986	0.927	**0.933**
CONt1+CONt2	0.979	0.881	0.912	0.907	0.987	0.922	0.910
ALLCONst	**0.983**	0.924	0.917	**0.916**	0.987	**0.929**	0.931

Table 5.4: *Transliteration detection results for different combinations of context-dependent DBN models. Values in bold indicate best performance compared to using transliteration similarity estimates 'individually'.*

manually designed methods on the cognate identification task. In this section, we have only explored a small space of combinations. There are many DBN model combinations that we can associate with the edit distance-based DBN modeling approaches and that should be worthy of investigation for improving transliteration detection quality.

5.4 Conclusion

In this chapter, we have applied a more generic DBN modeling approach in an experimental transliteration detection task. We successfully adapted three transduction-based DBN model templates associated with the approach to compute transliteration similarity. We specified eight transduction-based DBN models from the three DBN templates and evaluated them against the performance of the Pair HMMs using the same standard transliteration data for seven language pairs.

Results from our transliteration detection setup show that of the three transduction-based DBN model templates, the template for context-dependent DBN models results

in models that achieve the highest transliteration detection accuracy as compared to models based on the other transduction-based DBN templates. The context-dependent DBN models also achieve a considerable improvement in transliteration detection accuracy for all language pairs in comparison to the standard baseline approach of pair n-gram models. This includes the case for English-Russian where the Pair HMMs did not perform well as we would have expected. The context-dependent DBN models also result in better transliteration detection accuracy compared to Pair HMMs for 5/7 of the language pairs.

The transliteration detection results for the transduction-based DBN models in this chapter are generally similar to those that were reported for applying the edit distance-based models in pronunciation classification (Filali and Bilmes 2005) and cognate identification (Kondrak and Sherif 2006). An analysis of the transliteration detection results associated with the use of the DBN models also suggests that using ensembles of DBN models to compute transliteration similarity may guarantee improvements in accuracy.

It is important to note that although we have evaluated a reasonable number of transduction-based DBN models, the approach still offers a huge space of possible and feasible models which should be interesting to evaluate for computing transliteration similarity in a transliteration detection task.

Chapter 6

Applying DBN models in mining transliterations from Wikipedia

6.1 Introduction

In the previous two chapters, we evaluated the application of Pair Hidden Markov models (Pair HMMs) and transduction-based Dynamic Bayesian Network (DBN) models in an experimental transliteration detection task. The datasets that we used in the transliteration detection experiments comprised a collection of source and target language Named Entity (NE) pairs which we arbitrarily divided into reasonable training and test sets. We then used the trained edit distance-based DBN models and determined from the transliteration similarity scores, the rank of the candidate target NE (for each source NE in the test set). In the transliteration detection experiments, we assumed that there were no 'noisy' entities in the datasets. We also assumed knowledge about NEs in one language (which for convenience we refer to as a source language) for which we were supposed to determine a matching target language NE from a list of candidate target NEs in the test set for the target language. We refer to this as an ideal transliteration detection setup.

When we use real-world data to identify transliterations, we expect both the source and target language sides of the datasets to have 'noisy entities'. This is often the case in transliteration mining tasks where real-world data is obtained from Web-based comparable corpora including news corpora (Klementiev and Roth 2006, Sproat et al. 2006, Khapra et al. 2010) and encyclopedic corpora (mainly Wikipedia)

This chapter is an extended version of the following publications:
P. Nabende – Mining Transliterations from Wikipedia using Pair Hidden Markov Models, *Proceedings of the 2010 Named Entities Workshop*, pp. 76–80, July 2010, Uppsala, Sweden.
P. Nabende – Mining transliterations from Wikipedia using Dynamic Bayesian Networks, *To appear in Recent advances in Natural Language Processing*, Sept 2011, Hissar, Bulgaria.

(Kumaran et al. 2010b). This data is created by various users who represent it in various ways leading to high proportions of noisy entities. In this chapter, we aim to further establish the value of using the edit distance-based DBN models introduced in Chapters 4 and 5 for computing transliteration similarity by evaluating their application in mining transliteration pairs from Web-based noisy data.

In our investigation of the application of DBN models in this chapter, we hereby distinguish between two transliteration mining related tasks in which the DBN models are evaluated. In the first task, we follow the NEWS 2010 transliteration mining shared task setup (Kumaran et al. 2010b) where bilingual Wikipedia topics extracted using Wikipedia inter-language links (WILs) are provided as raw data for mining single word transliteration pairs. During the shared task, we applied the Pair HMM with distinct transition and emission parameters (Figure 4.4) to mine English-Russian transliteration pairs. After the shared task, we extended the evaluation of more DBN models on the English-Russian dataset and on the shared task datasets for the other four language pairs: English-Arabic, English-Chinese, English-Hindi, and English-Tamil.

In the second transliteration mining task, we investigate the application of Pair HMMs and context-dependent DBN models in mining single word transliteration pairs from Wikipedia article content in addition to mining from the topic pairs. In the next section, we continue with the descriptions of the two transliteration mining tasks. Later, we specify the edit distance-based DBN models we have chosen to evaluate in the two tasks of mining transliterations from Wikipedia.

6.2 Wikipedia – A source for transliteration mining

Wikipedia is an online encyclopedic resource in which over 270 languages have been used to write articles. In each language Wikipedia, there is a set of articles and for some articles access is provided to pages about the same topic in other languages. Figure 6.1 shows two Wikipedia articles about the same topic, one in English with the title "Arab Spring" whereas the other is in Arabic with the title الثورات العربية 'Arab Revolutions' and where the Arabic page has been accessed using the Arabic Wikipedia inter-language link which exists on the English page as shown. Currently, there is a growing interest in using such article pairs as comparable text for extracting various types of information. In our case, we use such article pairs across writing systems for mining single word transliteration pairs. As mentioned in the Introduction section above, we divide the Wikipedia transliteration mining task into two sub tasks. The most common task (which for convenience, we refer to here as the 'first' Wikipedia transliteration mining task) uses Wikipedia's link structure to mine bilingual NEs. Different types of links can be used for this task including (Erdmann et al. 2008):

Figure 6.1: *Two Wikipedia articles on the same topic but in different languages.*

inter-language links (an inter-language link is a link between two articles in different languages), redirect pages (a redirect page is a page that has no content but only a link to a target page), and link texts (these are the text part of the link that a user can click to reach a target page). For the NEWS 2010 shared task on transliteration mining (Kumaran et al. 2010b), only inter-language links were used to extract the topic pairs which were used as comparable text for mining single word transliteration pairs. We have also used the same datasets as those provided for the NEWS 2010 shared task on transliteration mining to evaluate the edit distance-based DBN models and we do not explore using the other types of links in this chapter.

In the 'second Wikipedia transliteration mining task', we use the main article content in addition to the topics. Although 'linked text' appears in main content, we do not distinguish it from the unlinked text. We therefore consider all named entities tagged in a pre-processing stage as candidate transliterations irrespective of whether they are part of linked text or not. As is depicted in Figure 6.1, it is clear that the main Wikipedia article content provides a lot more data that may lead to the mining of many transliteration pairs compared to the case of using only topic pairs.

6.2.1 Transliteration mining using Wikipedia inter-language links

Recently, many studies have found it inexpensive to extract a lot of specific cross-language data including named entities and terminologies by using only Wikipedia's inter-language links (Adafre and de Rijke 2006, Mohammadi and GhasemAghaee 2010, Kirschenbaum and Wintner 2010, Erdmann et al. 2008). We will first use comparable Wikipedia topics that have been identified from the Inter-language links to constitute the collection of raw data to which the DBN models will be applied and evaluated for mining single word transliteration pairs.

Figure 6.2 is an overview of the the NEWS 2010 transliteration mining shared task setup. We use the same setup used in the shared task to simplify our comparison of the application of the DBN models against state-of-the-art approaches that were evaluated (Kumaran et al. 2010a) on the same standard transliteration corpora. In Figure 6.2, the 'Interwiki links' refer to the collection of article titles such as the English and Arabic titles in Figure 6.1 that are provided for the shared task as-is. Since manual extraction of transliteration pairs is used to create the Gold standard, only a small random sample of the many noisy article titles are provided as the gold

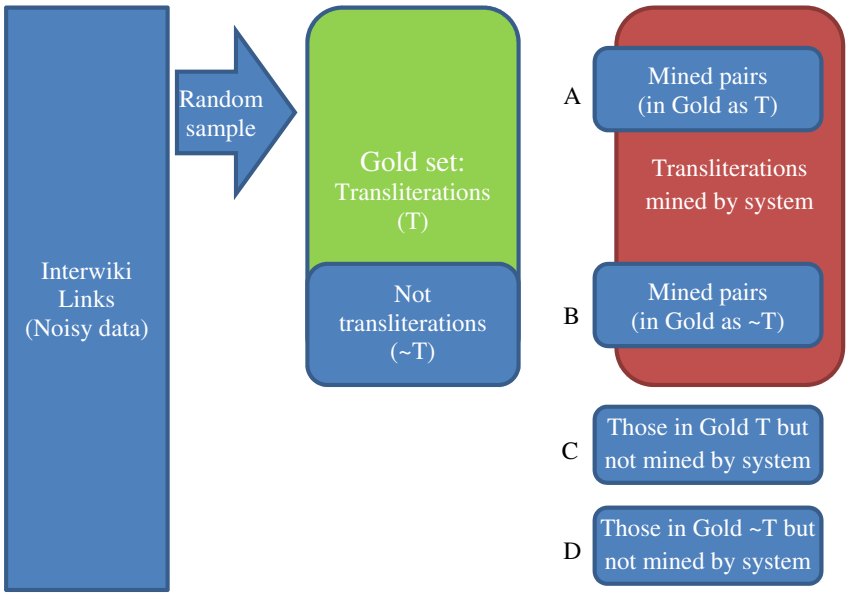

Figure 6.2: *Transliteration mining task and evaluation overview following the 2010 NEWS shared task. (Adapted from Kumaran et al., 2010b).*

set. Given the raw interwiki links, transliteration mining systems that participated in the shared task were supposed to return single word transliteration pairs from each pair of Wikipedia topics if they identified any, and otherwise, returned nothing if no transliteration pairs were identified. On the evaluation side of Figure 6.2, A refers to pairs from the transliteration mining system that are evaluated as True Positives (TP), B refers to those evaluated as False Positives (FP), C refers to those evaluated as False Negatives (FN), and D to those evaluated as True Negatives (TN). In section 6.4, we will refer to the notations in Figure 6.2 when describing the evaluation setup for this transliteration mining task.

6.2.2 Transliteration mining using comparable Wikipedia article text

In addition to using Wikipedia's article topics to mine transliterations, we also explore the use of the main article content. The motivation here as suggested in Figure 6.1 is quite obvious, that we expect to find more named entities in the main article content than there are in the article titles. However, for the same number of Wikipedia topics, we expect the datasets to grow exponentially hence requiring more time and effort to pre-process and analyse. Instead of using all the article pairs between two languages for mining transliterations, we limited the size and identified only article pairs that we hypothesized to contain a sufficient number of transliteration pairs. To help us achieve that, we used as a starting point existing statistics about the number of page visits associated with the English Wikipedia for a given duration. Specifically, we used statistics about the English Wikipedia articles that got the highest number of page hits during the month of august 2009 and chose those that ranked highly in that regard. The corresponding articles in the other language were then easily retrieved through the inter-language links. Because there is need to manually annotate a gold standard from this type of data, we limited the number of document pairs to enable the evaluation of several DBN models for detecting transliteration pairs. Also, because of my own lack of expertise in most of the writing systems, we evaluated the DBN models for the English-Russian language pair only.

6.3 DBN model selection for transliteration mining

The DBN models that we apply to compute transliteration similarity in the transliteration mining tasks in this chapter have already been introduced in Chapters 4 and 5. In this section we present the specific DBN model structures we chose to apply in mining transliterations from Wikipedia. To explain the choice of model structures, we briefly review the results from the transliteration detection experiments in

Chapters 4 and 5. Chapter 5 results show that the context-dependent DBN models outperform the other DBN models in most of the language pairs. It is therefore natural to consider them as the most suitable from the set of DBN models for application in mining transliterations from Wikipedia. However, Chapter 5 results also show that the best performing Pair HMMs achieved accuracies and MRRs comparable to those for some of the context-dependent transduction-based DBN models. The Pair HMMs even performed best in the detection of English-Hindi and English-Kannada transliterations. Moreover, we also note that the application of the transduction-based DBN models was limited by slow runtimes during inference for training the models and for computing transliteration similarity. This drawback on the use of the transduction-based DBN models could prove costly in a task involving the analysis of huge amounts of data. This is indeed the case in the transliteration mining tasks in this chapter where we apply the models to mine from hundreds of thousands of Wikipedia paired topics.

The transliteration mining tasks in this chapter necessitate an evaluation of most of the Pair HMMs and transduction-based DBN models that were introduced in Chapters 4 and 5. However, the results from Chapters 4 and 5 suggest that the use of some algorithms (for the case of Pair HMMs) and some model generalizations (for the case of transduction-based DBN models) always lead to gains in transliteration detection quality over the other Pair HMM algorithms and transduction-based DBN model generalizations. Therefore, we have chosen to consider model structures from both DBN approaches that achieved the highest transliteration detection accuracy and MRR for any of the language pairs investigated in this chapter. In the following two subsections, we specify the chosen DBN model structures for each approach.

6.3.1 Pair HMMs

From the Pair HMM approach, we use the Pair HMM structures and the respective algorithms that resulted in the best transliteration detection accuracy in Chapters 4 and 5. The Forward log-odds algorithm for the Pair HMM with three transition parameters (which we denote here as PHMM3 – see Figure 4.3) led to the highest transliteration detection accuracy for most of the language pairs compared to other Pair HMM structures and algorithms. However, the other scoring algorithms for PHMM3 resulted in lower transliteration detection accuracy compared to the algorithms for the other Pair HMM variants where transition parameters were used in computing transliteration similarity. We therefore decided to evaluate at least two Pair HMM variants that had the highest transliteration detection accuracy for a particular language pair. Based on that, we chose to use PHMM3 and the Pair HMM variant that uses five transition parameters (denoted here as PHMM5 – see

Figure 4.2) with their respective forward log-odds algorithms for mining single word transliteration pairs from English-Chinese and English-Tamil Wikipedia topic pairs. For English-Hindi, we use PHMM3 and the Pair HMM variant that uses nine distinct transition parameters (denoted here as PHMM9 – see Figure 4.4). For English-Russian, we use PHMM5 and PHMM9.

6.3.2 Transduction-based context-dependent DBN models

From the transduction-based DBN model generalizations presented in Chapter 5, we also chose the best performing DBN models. With reference to the transliteration detection results in Chapter 5, the context-dependent DBN models led to the highest transliteration detection accuracy and MRR compared to other transduction-based DBN models on all transliteration datasets. We have therefore chosen to evaluate only context-dependent DBN models for computing transliteration similarity in mining transliterations from Wikipedia. Table 6.1 is a summary of the selected Pair HMM and context-dependent DBN models.

We use a context-dependent DBN model that incorporates the dependency of the edit operation variable on the current and previous target string character (denoted here as CONt2) for computing transliteration similarity for four language pairs: English-Arabic, English-Chinese, English-Hindi, and English-Tamil. We have chosen to use CONt2 for English-Arabic since it led to better transliteration detection accuracy for most of the language pairs compared to the other DBN models in Chapter 4. For English-Russian, we use a context-dependent DBN model that models a dependency of the edit operation on the current source character only (See section 5.2.3 for detailed description of the transduction-based DBN model generalizations). Note that here, we also take English to be the target language and each of the other languages to be the source language. Therefore, CONt2 means that we change and

Language pair	PHMMm1	PHMMm2	CON DBN
English-Arabic	PHMM3 forwLO	PHMM9 forwLO	CON(s2&t2)
English-Chinese	PHMM3 forwLO	PHMM5 forwLO	CON(s2&t2)
English-Hindi	PHMM3 forwLO	PHMM9 forwLO	CON(s2&t2)
English-Russian	PHMM5 forwLO	PHMM9 forwLO	CON(s1&t1)
English-Tamil	PHMM3 forwLO	PHMM5 forwLO	CON(s2&t2)

Table 6.1: *PHMMs and transduction-based DBN models for mining transliterations from Wikipedia. forw. LO refers to the forward log-odds algorithm for a given Pair HMM. PHMMm1 and PHMMm2 refer to the PHMMs that have the best and second best accuracy and MRR from Chapter 4 results.*

consider English as the source language and each of the other languages as the target language.

6.4 Experiments using NEWS 2010 shared task setup

6.4.1 Wikipedia inter-language link data

In this task, three sets of data were provided per language pair for the NEWS 2010 shared task on transliteration mining (Kumaran et al. 2010b). For each language pair, 1000 hand picked pairs of single word NEs were provided as seed data for training any proposed transliteration mining system. Then pairs of corresponding article topics that had been collected using Wikipedia Inter-language links (WILs) from a given English Wikipedia data dump were provided as data from which single-word transliteration pairs were to be mined. For all language pairs, the Wikipedia paired topics were noisy with a large number of unwanted symbols which necessitated a data pre-processing step before applying any transliteration mining system. Most of the unwanted symbols included: temporal and numerical expressions, punctuation symbols, formatting symbols, mathematical operator symbols, and characters from other writing systems that are not part of the source or target language writing system. 1000 Wikipedia links were also randomly selected (from the large noisy WILs) from which single word NE pairs for each link were manually identified and annotated to constitute the gold standard set used to evaluate the application of the transliteration mining systems. The gold set also had cases where there existed no transliteration pairs.

For this particular task, we expect there to be no effect on the final result in using one language as the source language and the other language as the target and vice-versa since we do not have prior knowledge about either the source or target string. However, as in the transliteration identification tasks in Chapters 4 and 5, we use English on the target side whereas each of the other languages is used on the source side.

To reduce data sparseness, all datasets were converted to lowercase. For each of the DBN approaches, the source and target language alphabets used by the models were restricted to the characters that were found in the seed set for a first run of the models. Although the seed sets contained the standard characters for most of the languages, there were a lot of characters (mostly diacritics) in the noisy Wikipedia topics that were missing in the seed set. With the aim of determining how to evaluate words in the noisy set with the 'new' characters, we checked on the reference set and observed that the words in the gold standard data hardly used any characters apart from those in the seed set. We expect that an additional benefit of the transliteration

mining task is to find new words including those that use uncommon characters which usually convey an unusual original pronunciation of the word. Thus, we also considered words with 'new' characters irrespective of how they were treated in the evaluations of the NEWS 2010 transliteration mining shared task (Kumaran et al. 2010b). Table 6.2 is a summary of the datasets indicating the total number of noisy Wikipedia paired topics for each language pair, and the final number of Wikipedia topic pairs that were used to build the set of candidate NEs per language pair.

Language pair	before pre-processing	after pre-processing
English-Arabic	90926	85244 (93.8%)
English-Chinese	196047	175013 (89.3%)
English-Hindi	16963	14620 (86.2%)
English-Russian	345969	296053 (85.6%)
English-Tamil	13883	13249 (95.4%)

Table 6.2: *Number of Wikipedia topic pairs before and after pre-processing.*

We transformed each dataset into numerical representation to enable efficient computation using the training and transliteration similarity estimation algorithms for the edit distance-based DBN models. For words that had 'new' characters in the Wikipedia topics, we used only information about the 'known' characters and ignored the 'unknown' characters when computing transliteration similarity.

Acquiring additional training pairs - Iterative training

One critical issue in learning a transliteration model is the use of manually labeled training data. To ensure accurate learning, a large number of transliteration matches are needed. But for the shared task, only few are provided as seed data. One approach we use to automatically acquire additional unverified training pairs from the noisy Wikipedia data is to bootstrap from a seed set. This is described in the following paragraph.

Assuming that the noisy Wikipedia data constitutes our set of unlabeled data U. We first utilize the seed set to learn a transliteration model which for convenience we denote here by T_1. We apply T_1 to U to identify the set of transliteration pairs L_1 whose similarity score according to the transliteration model are above a selected threshold score th_1. We then add L_1 to S and use $(S + L_1)$ to learn a new transliteration model which is applied to U again to identify an additional set of transliteration pairs L_2 whose similarity score according to the transliteration model is above a selected threshold value th_2. Following this approach we evaluate the performance of

the resulting models for a given number of iterations. At each iteration, we test the learned transliteration model on the same gold standard set G.

6.4.2 Evaluation setup and results

The Pair HMM approach and the transduction-based DBN modeling approach are both generation-based approaches[1] whereby, for each candidate transliteration pair, transliteration similarity is computed based on an underlying process which is perceived to generate the two words. Then, based on some criterion (which in our case is a *threshold* value), a decision is made as to whether the candidate transliteration pair should be regarded by the transliteration system as a suggested true transliteration pair or not. In the following, we use a sample from the English-Russian Wikipedia topic pairs to demonstrate the evaluation setup used for the 2010 shared task on transliteration mining.

Table 6.3 shows the sample of English and Russian Wikipedia topics from which single word transliteration pairs are to be mined. Each underlined word in the Table has a transliteration match in the other language. Given the Wikipedia topics in Table 6.3, we expect the mining system to find transliterations pairs as illustrated in Table 6.4. Based on the sample results in Table 6.4, there are some important points to note with respect to the task. First, we see that not all Wikipedia article topics in Table 6.3 will result in a transliteration pair. This is illustrated in Table 6.4 where we use <NULL> to indicate that no transliteration pairs were found in some topic pair. Secondly, following the NEWS 2010 shared task setup (Kumaran et al. 2010a), morphological variations on either side are not regarded as transliterations. Thirdly,

#	English Wikipedia title	Russian Wikipedia title
1	Chereksky District	Черекский район Кабардино-Балкарии
2	Oleksandr Palyanytsya	Паляница, Александр Витальевич
3	Erzerum Offensive	Эрзурумское сражение
4	Mauricio Funes	Фунес, Маурисио
5	Proper equilibrium	Собственное равновесие

Table 6.3: *Sample of English-Russian Wikipedia topic pairs. The underlined words indicate that they have a transliteration match in the other language.*

[1] The other common category of approaches that were evaluated in the NEWS 2010 shared task on transliteration mining (Kumaran et al. 2010b) are discriminative approaches which treat the mining task as a binary classification problem. The discriminative approaches necessitate the construction and use of a classifier to decide whether a candidate pair of words qualifies as a transliteration pair or not.

#	English name	Russian name
1	Chereksky	Черекский
2	Oleksandr	Александр
2	Palyanytsya	Паляница
3	<NULL>	<NULL>
4	Mauricio	Маурисио
4	Funes	Фунес
5	<NULL>	<NULL>

Table 6.4: *Sample of expected transliteration mining result from the raw Wikipedia topics of Table 6.3.*

following the common definition for transliteration, translations having distinct phonetic transcriptions (a feature needed to qualify them as transliteration pairs) on either side are also not considered as transliterations. For example, although 'District' and 'район' in the first Wikipedia topic pair in Table 6.3 can be taken as translations of each other since they refer to a similar entity (pertaining to territorial administrative divisions) in the respective languages, their respective phonetic transcriptions /dɪstrɪkt/ and /raion/ (using the International Phonetic Alphabet (IPA)) are very different implying that they can not be treated as transliteration pairs.[2] Finally, before training the models, we first convert the datasets (i.e. in the case of English and Russian) that have uppercase characters to lowercase.

a) Evaluation metrics

We train each Pair HMM using the Baum-Welch algorithm (Chapter 4), and each context-dependent DBN model using a generalized expectation maximization algorithm (Chapter 5). We then use the trained models to compute transliteration similarity between candidate transliteration pairs as described above. After applying each model, we check a subset of the transliteration similarity scores to enable the specification of different threshold values. If we specify a low threshold value, we get many suggestions of potential transliteration pairs returned by the mining system. If we specify a high threshold value, we get very few suggestions returned by the mining system. The collection of transliteration pairs whose transliteration similarity scores beat a given threshold value for each language pair are then evaluated against the gold set which contains a random selection of manually annotated topic pairs from

[2]A true Russian transliteration for the English word 'District' is 'Дистрикт' and it is used to mean the same entity as in English. Conversely, a true English transliteration for the Russian word 'район' is 'raion' and it is also used to mean the same entity.

the initial set of topic pairs.

We evaluate each edit distance-based DBN model for mining single word transliteration pairs from Wikipedia topic pairs using three related metrics as described for the NEWS 2010 shared task on transliteration mining in Kumaran et al. (2010b, 2010a). The evaluation metrics are: Precision, Recall, and F-score[3]. With reference to Figure 6.2, we compute the measures as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{6.1}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6.2}$$

$$\text{F-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6.3}$$

where:

- TP (True Positives) are word pairs that are identified as correct transliterations by the transliteration mining system and indeed are in the gold standard.
- FP (False Positives) are word pairs that are identified as correct transliterations but are incorrect transliterations according to the gold standard.
- FN (False Negatives) are word pairs that are identified as incorrect transliterations but are actually correct transliterations according to the gold standard.
- TN (True Negatives) are word pairs that are identified as incorrect transliterations and are indeed incorrect transliterations as per the gold standard. As seen in the expressions above we do not use these in the computations for *Precision* and *Recall* even when they are part of the result.

The different threshold values as introduced above enable us to plot Precision-Recall curves for a more general analysis of the transliteration mining results. However, when comparing the methods on single Precision, Recall, and F-score values, we set a threshold value that we think will result in an optimal discrimination between true transliteration pairs and incorrect transliteration pairs. In this case, We analyse a subset of the results returned by the system and then identify true transliteration pairs whose similarity scores we use to subjectively set the threshold value.

[3]We use the F_1 score in our evaluations with respect to use for evaluation in the 2010 NEWS shared task on transliteration mining. But since, the aim of a transliteration mining task is to acquire only correct transliterations, it is more appropriate to adapt this measure so that more importance is attached to *Precision* than *Recall*

b) English-Russian transliteration mining results

Table 6.5 shows the first 10 results out of a total of 15 for the transliteration mining methods that were evaluated on the English-Russian dataset. During the shared task we applied a Pair HMM with nine distinct transition parameters and distinct emission parameters (Figure 4.4). We use PHMM9F to denote this Pair HMM in Table 6.5. When applying PHMM9F, we used the forward algorithm to compute transliteration similarity. In Table 6.5, we have also included the result for a context-dependent DBN model which models a dependency of the edit operation variable on the current target character (CONt1). Since we did not apply the context-dependent DBN model during the shared task, its result is initially not included in the shared task report on the English-Russian dataset. However, we applied it to the same English-Russian datasets as used during the shared task. Table 6.5 specifies two types of runs: a *standard run* where only the seed datasets are supposed to be used to train the transliteration mining systems, and a *non-standard run* where participants are allowed to use additional and / or external data to complement the seed datasets. The results in Table 6.5 alone illustrate the varied applicability of the HMM framework. These results also suggest a comparable performance of the HMM-based methods to other state-of-the-art methods. The best method, NED+ (which is an extended form

Run type	Description	Precision	Recall	F-Score
**Standard	CONt1	0835	0.815	0.825
*Standard	PHMM9F	0.780	0.834	0.806
Standard	NED+	**0.880**	**0.869**	**0.875**
Standard	HMM + PC	0.813	0.839	0.826
Non-standard	LFS + Seed$^+$	0.797	0.853	0.824
Standard	StringKernel	0.746	0.889	0.811
Standard	HMM	0.868	0.748	0.804
Standard	HMM + PC + IterT	0.843	0.747	0.792
Non-standard		0.730	0.870	0.790
Standard	DirecTL+	0.778	0.795	0.786

Table 6.5: *Results of the NEWS 2010 transliteration mining shared task on English and Russian data (Source: Kumaran et al. (2010b)). During the NEWS 2010 shared task on transliteration mining, we applied PHMM9F which is the Pair HMM variant with distinct emission and transition parameters. Here, we used the forward algorithm to compute transliteration similarity. The two asterisks (**) indicate that the result for CONt1 (a context-dependent DBN model) was not included in the shared task report since it was evaluated at a later time after the shared task.*

of the normalized edit distance (NED) measure) is reported by Jiampojamarn et al. (2010) to be their baseline method. NED is defined as the ratio of the edit distance between two strings and the maximum length of the two strings. The relatively good performance of the NED+ measure and the HMM-based methods suggests that there may be no gain in using more complex methods to model transliteration similarity between English and Russian. The use of a context-dependent DBN model led to an improved F-score value over the Pair HMM approach but still below that for the simple NED+ approach.

c) English-Hindi transliteration mining results

For English-Hindi, we applied five Pair HMMs and two context-dependent DBN models. Figure 6.3 shows the Precision-Recall curves for the different models. In Figure 6.3, PHMM9F denotes the Pair HMM with distinct transition and emission parameters (Figure 4.4) and where the Pair HMM forward algorithm is used to compute transliteration similarity. PHMM9 refers to the same Pair HMM denoted by PHMM9F but where we use the Pair HMM forward log-odds algorithm to compute transliteration similarity. PHMM9IterT refers to the same Pair HMM as PHMM9 but which we train iteratively as described in section 6.4.1 on "acquiring additional training data", and where we use the forward log-odds algorithm to compute transliteration similarity. PHMM3 refers to the Pair HMM with three transition parameters between the edit states and where we use the Pair HMM forward log-odds algorithm to compute transliteration similarity. PHMM3IterT refers to the same Pair HMM as PHMM3 but which we also train iteratively and where we also use the forward log-odds algorithm to compute transliteration similarity.

Figure 6.3 shows that the Pair HMMs where the forward log-odds algorithm is used to compute transliteration similarity (that is PHMM3, PHMM9, PHMM3IterT and PHMM9IterT) achieve a superior performance as their curves are closer to the upper right corner of the graph (where Precision and Recall is maximized) compared to the curves for PHMM9F and the two context-dependent DBN models.

This result is similar to the transliteration detection result for the English-Hindi language pair in Chapter 5 where a Pair HMM using the forward log-odds algorithm outperformed the context-dependent DBN models. The transliteration mining result in this case suggests that the approach of Pair HMMs could be suitable for mining transliteration English-Hindi transliteration pairs.

In Table 6.6, we present our results for the Pair HMMs and context-dependent DBN model alongside the first five shared task results for the approaches that were evaluated in mining English-Hindi transliteration pairs. The notations for the Pair HMMs and context-dependent DBN models are the same as defined above.

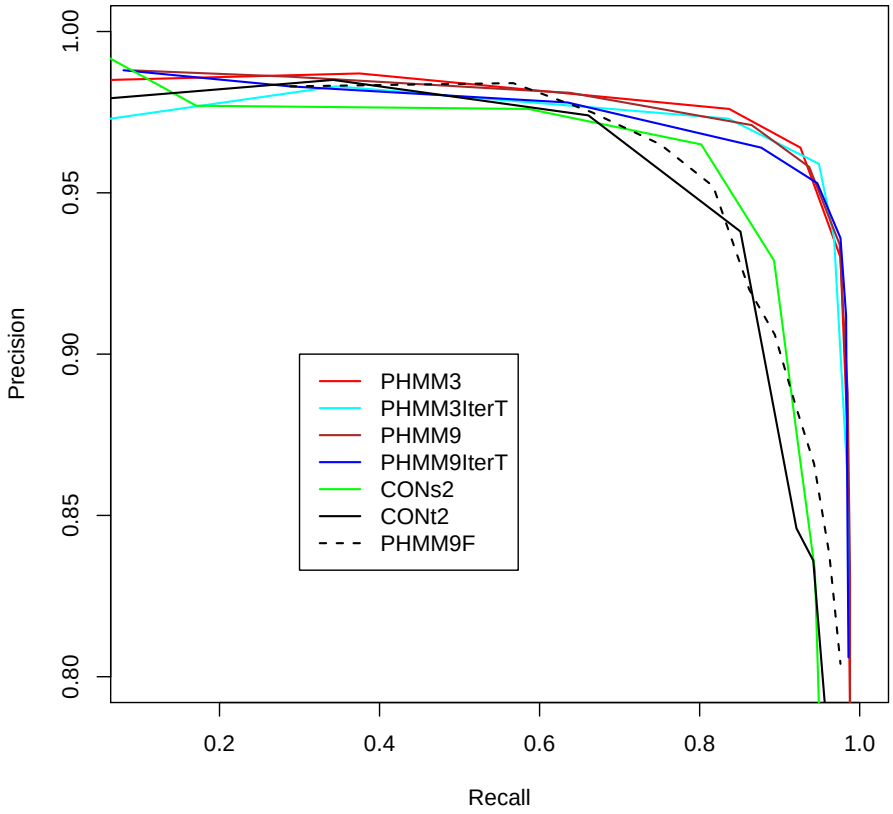

Figure 6.3: *Precision-Recall curves for Pair HMMs and context-dependent DBN models after evaluation on 982 English-Hindi test items at different threshold values. CONs2 (respectively CONt2) is a transduction-based DBN model where there is a dependence of the edit operation variable on the current and previous characters in the source word (respectively target word). PHMM9F refers to the Pair HMM with distinct transition parameters (PHMM9) where we use the forward algorithm to compute transliteration similarity, PHMM9 refers to PHMM9 where we use the forward log-odds algorithm. PHMM9IterT refers to the Pair HMM that we trained iteratively and used the forward log-odds algorithm. PHMM3 refers to the Pair HMM with three transition parameters where we use the forward log-odds algorithm, and PHMM3IterT refers to the PHMM3 where we train the model iteratively.*

Run type	Description	P	R	F-score
* Standard	PHMM3_FLO	**0.930**	**0.976**	**0.952**
* Standard	PHMM3IterT	**0.959**	**0.949**	**0.954**
* Standard	PHMM9_FL0	**0.934**	**0.975**	**0.954**
* Standard	PHMM9IterT	**0.936**	**0.976**	**0.955**
* Standard	CONs2	0.929	0.893	0.911
* Standard	CONt2	0.938	0.851	0.893
Standard	StringKernel	0.954	0.895	0.924
Standard	NED$^+$	0.875	0.941	0.907
Standard	DirecTL$^+$	0.945	0.866	0.904
Standard	(HMM+PC+IterT)+PC	0.953	0.855	0.902
Standard	BK-2007	0.883	0.880	0.882

Table 6.6: *English-Hindi transliteration mining results for Pair HMMs and context-dependent DBN models against NEWS 2010 Shared task results (Kumaran et al. 2010b). P refers to Precision whereas R refers to Recall. We again use an asterisk to indicate that we evaluated the respective models later after the shared task but on the same dataset.*

As Table 6.6 shows[4], the Pair HMMs where we use the forward log-odds algorithm to compute transliteration similarity result in a considerably better F-score value compared to that for the best performing approach in the shared task and for the other edit distance-based DBN models. This result is quite surprising considering that we use the Pair HMMs in their most basic form. Although iterative training of the models (that is for PHMM3IterT and PHMM9IterT) leads to the best F-score values, there is only a slight increase over the case where we do not iteratively train for the same model structures and where we use the same scoring algorithm (that is for PHMM3 and PHMM9). Probably, the already high F-score values will be hard to improve using the same model structures with the same algorithms.

d) English-Tamil transliteration mining results

For English-Tamil, we present the results for only four models: two Pair HMMs and two context-dependent DBN models. The Pair HMMs (PHMM3 and PHMM5) and the context-dependent DBN models (CONs2 and CONt2) are as defined in the previous subsection. Figure 6.4 shows the Precision-Recall curves for the four models. In Figure 6.4, we again see that the Pair HMMs achieve a superior performance as

[4]Note that the results for the Pair HMMs and context-dependent DBN models in Table 6.6 were not part of the NEWS 2010 transliteration mining shared task evaluation. Although we used the same datasets as those provided for the shared task, we conducted transliteration mining experiments at a later time.

their curves are closer the upper right corner of the graph compared to the curves for the context-dependent DBN models. The difference between the curves for the Pair HMMs and the context-dependent DBN models in Figure 6.4 is even bigger compared to the difference in Figure 6.3 for the English-Hindi transliteration mining results.

In Table 6.7, we present transliteration mining results for the Pair HMMs and the context-dependent DBN models alongside the first five shared task results for approaches that were evaluated in mining English-Tamil transliteration pairs. As Table 6.7 shows, the Pair HMMs again result in considerably better F-score values

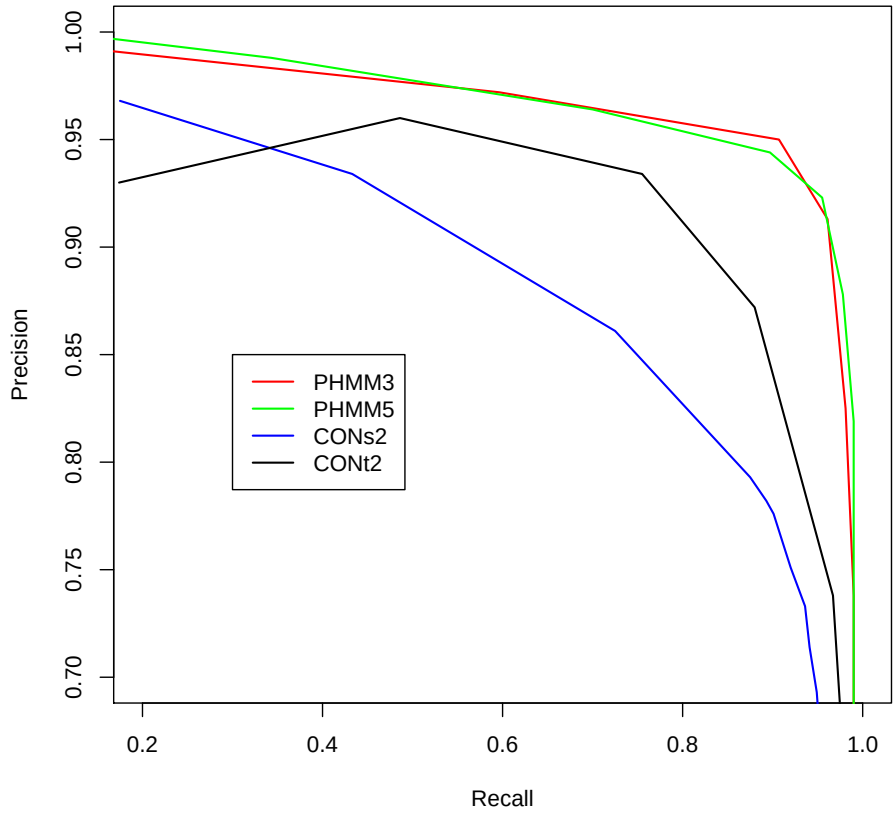

Figure 6.4: *Precision-Recall curves for Pair HMMs and context-dependent DBN models after evaluation on 690 English-Tamil test items at different threshold values.*

Run type	Description	P	R	F-score
* Standard	PHMM3	**0.913**	**0.966**	**0.936**
* Standard	PHMM5	**0.923**	**0.955**	**0.939**
* Standard	CONs2	0.782	0.893	0.834
* Standard	CONt2	0.872	0.880	0.876
Standard	StringKernel	0.923	0.906	0.914
Non-Standard	LFS + Seed$^+$	0.910	0.897	0.904
Standard	LFS	0.899	0.814	0.855
Standard	LFS	0.913	0.790	0.847
Standard	BK-2007	0.808	0.852	0.829

Table 6.7: *English-Tamil transliteration mining results for Pair HMMs and context-dependent DBN models against NEWS 2010 shared task results (Kumaran et al. 2010b).*

compared to the best performing approach in the shared task. One of the context-dependent DBN models (CONt2) achieves a competitive F-score value although not better than the best shared task result.

e) English-Chinese transliteration mining results

For English-Chinese, we applied two Pair HMMs (PHMM3 and PHMM5) and two context-dependent DBN models (CONs2 and CONt2). We used the forward log-odds algorithm to compute transliteration similarity using PHMM3 and PHMM9. The context-dependent DBN models are also as defined for the previous language pairs. Table 6.8 shows the transliteration mining results from using the Pair HMMs and context-dependent DBN models for computing transliteration similarity alongside results for the approaches used during the NEWS 2010 shared task.

As Table 6.8 shows, none of the edit distance-based DBN models achieve an F-score value higher than that for the best shared task result. However, considering the general poor performance in mining English-Chinese transliteration pairs, the F-score values for the edit distance-based DBN models are very competitive. The transliteration mining results in Table 6.8 for the edit distance-based DBN models are slightly different compared to English-Hindi and English-Tamil transliteration mining results in the two previous subsections. Here, although the best transliteration mining result is from a Pair HMM, we see a mixed performance. Unlike the case for English-Hindi and English-Tamil, we see that CONt2 outperforms one of the Pair HMMs and it achieves an F-score value comparable to that for the best Pair HMM result. It should also be noted that unlike the case for the other language pairs, we used an intermediate representation step for English-Chinese data. That is, we first transcribed Chinese characters to characters of the Latin alphabet using the Pinyin

Run type	Description	P	R	F-score
* Standard	PHMM3	0.574	0.337	0.425
* Standard	PHMM5	0.526	0.335	0.410
* Standard	CONs2	0.435	0.241	0.310
* Standard	CONt2	0.546	0.334	0.414
Standard	Matching	**0.698**	**0.427**	**0.530**
Non-Standard		0.700	0.430	0.530
Standard	(HMM+IterT)+PC	1.000	0.030	0.059
Standard	HMM+IterT	1.000	0.026	0.050
Standard	HMM+PC	1.000	0.024	0.047
Standard	(HMM+PC+IterT)+PC	1.000	0.022	0.044
Standard	HMM	1.000	0.016	0.032
Standard	HMM+PC+IterT	1.000	0.016	0.032
Standard	DirecTL+	0.045	0.005	0.009

Table 6.8: *English-Chinese transliteration mining results for Pair HMMs and context-dependent DBN models against NEWS 2010 shared task results (Kumaran et al. 2010b).*

Romanization system. Transliteration mining results from a preliminary run where we applied the edit distance-based DBN models on Chinese data in its original orthography showed very low F-score values and we do not report them here. We suspect that the lack of applying a segmentation step for some of the Chinese topics could have considerably affected transliteration mining qaulity in our experimental runs. Nonetheless, the English-Chinese transliteration mining results suggest that the edit distance-based DBN models could also be valuable for languages that use relatively complex writing systems.

f) English-Arabic transliteration mining results

For English-Arabic, we report on transliteration mining results from applying three Pair HMMs (PHMM3, PHMM5, and PHMM9) and two context-dependent DBN models (CONs2 and CONt2). We use the forward log-odds algorithm to compute transliteration similarity for the Pair HMMs. All the edit distance-based DBN models are also as defined for the other language pairs in the previous subsections. Table 6.9 shows the transliteration mining results for the edit distance-based DBN models with the NEWS 2010 transliteration mining shared task results.

As table 6.9 shows, none of the edit distance-based DBN models leads to an F-score value better than that of the best shared task result. Although the F-score values for the edit distance-based DBN models are within the range of F-score values

Run type	Description	P	R	F-score
* Standard	PHMM3	0.794	0.877	0.833
* Standard	PHMM5	0.821	0.860	0.840
* Standard	PHMM9	0.816	0.876	0.845
* Standard	CONs2	0.752	0.740	0.746
* Standard	CONt2	0.835	0.739	0.784
Standard	FST model	**0.887**	**0.945**	**0.915**
Standard	FST model	0.859	0.952	0.903
Standard	Phonetic distance	0.923	0.830	0.874
Standard	HMM+IterT	0.886	0.817	0.850
Standard	HMM+PC	0.900	0.796	0.845
Standard	(HMM+IterT)+PC	0.818	0.827	0.822
Standard	(BK-2007)	0.834	0.798	0.816

Table 6.9: *English-Arabic transliteration mining results for two Pair HMMs and two context-dependent DBN models with the NEWS 2010 shared task results (Kumaran et al. 2010b).*

that were reported for the approaches that participated in mining English-Arabic transliteration pairs during the shared task, the difference in F-score between the best shared task result and the best performing DBN model is larger compared to differences for the other language pairs where the DBN models did not beat the best shared task result.

The transliteration mining results in Table 6.9 for the edit distance-based DBN models correspond with the English-Hindi and English-Tamil transliteration mining results. In Table 6.9, we see that the Pair HMMs outperform the context-dependent DBN models which is the case for English-Hindi and English-Tamil. However, the difference in F-score for the English-Arabic case is relatively higher than the differences for the English-Hindi and English-Tamil case. For the three language pairs (English-Arabic, English-Hindi and English-Tamil), the transliteration mining results suggest that Pair HMMs could be better for computing transliteration similarity in the process of mining transliterations from noisy data in comparison to using context-dependent DBN models. Since we have evaluated only two context-dependent DBN models in each case, the conclusions we make at this point regarding a comparison of the Pair HMMs and the context-dependent DBN models for mining transliteration pairs for the three language pairs may be insufficient. Instead, it should be interesting to evaluate other transduction-based DBN models to determine whether they could lead to improvements in transliteration mining quality for the three language pairs

over the context-dependent DBN models and hence that would contradict transliteration detection results in Chapter 5 where context-dependent DBN models performed best.

6.5 Experiments using comparable article content

6.5.1 Extracting training data from Wikipedia

For this set of experiments, we use Wikipedia's topic pairs to automatically acquire a training set. Instead of searching for every possible topic pair, we restricted the scope of search to named entities, specifically person names, where we expect to find correct transliteration matches. We extracted name pairs by exploiting the structured nature of information in Wikipedia info-boxes (Bouma et al. 2009). We searched for entries that match a given pattern with respect to a language or country in Wikipedia categories such as "citizenship", "nationality", and "place of birth", and then extracted the corresponding titles in the other language using inter-language links which are on the same Wikipedia page. Despite this restricted search, there were differences in the representation of names between the two languages and hence necessitating data pre-processing steps. Firstly, we observed a difference in the order of names. While most names in English start with the 'first name' (e.g. Barack Obama), many corresponding names in Russian start with a second name followed by a comma and lastly the first name (e.g. Обама, Барак). Here, we simply changed the order by swapping the strings preceding and following a comma for all the Russian names. Secondly, we observed the presence of middle names in some topics. Here, we removed any abbreviations for middle names and considered only the first and last names. We used only the last name if the number of names was different between the source and target language topics and there was only one name in one of the languages. Lastly, some names had a hyphen in one (for example Shin-ichiro and Синъитиро) or both topics (for example Cary-Hiroyuki and Кэри-Хироюки). Names that had a hyphen in both topics were used as provided, but names that had a hyphen in only one topic (for example Chung-hee and Хи) were filtered out. After pre-processing, we had 3142 corresponding English-Russian names that we designated as training data.

6.5.2 Comparable Wikipedia article content data

There are various criteria we can use to identify cross language Wikipedia article content for mining transliterations. However, it is important to select article pairs where we would expect to mine a reasonable number of transliteration pairs. We can consider, as a plausible premise, that a Wikipedia article with a large number of words

has a large number of named entities. We can then evaluate its corresponding article in the other language to decide whether there is also a sufficient number of words from which we expect to get a large number of candidate named entities. Here, we would consider every entity in the content regardless of how the content is structured. Given the two comparable Wikipedia articles, an estimation of the content size in them can lead us to a decision of whether to use them for mining transliteration pairs or not. This approach to extracting comparable Wikipedia articles requires a search through the whole database and performing a ranking with the aim of selecting article pairs with the highest number of words.

A different approach that we use in selecting Wikipedia article pairs for mining transliteration pairs, is to follow clues from Wikipedia's statistics about articles. In our case, we use statistics associated with the most accessed English Wikipedia pages in a given month. We expect that a page that is visited very often, not only generates a lot of interest but also has a big content size. We also conjecture that the 'interest' associated with an article leads to corresponding representations about the same topic in other language Wikipedias. We also make a rough conclusion that its size in the other language should be reasonably large as well. Using statistics about the number of page hits per month for the English Wikipedia articles during the year 2009, we identified some 10 articles that had the highest number of page hits per day[5] for the month of august 2009. We retrieved the corresponding Russian articles through the Inter-language links on the same page.

Given the English and Russian Wikipedia articles, we identified various written entities that were not relevant for transliteration mining. The irrelevant entities here include: temporal and numerical expressions, entities using characters from other writing systems that are different to the writing system for source or target language, punctuation symbols, and different formal expressions such as mathematical expressions. In the following we present the steps taken to filter out the irrelevant entities. First, we extracted only those words that were written using the Latin alphabet for English and the Cyrillic alphabet for Russian. A simple regular expression for this purpose is sufficient for removing most of the irrelevant entities. Secondly, we note that it may not be useful to consider every possible word from each of the articles as a candidate named entity. Instead, it could be more helpful if we identified types of words in the article pairs for which transliteration is commonly used. Named entities are well known to constitute the highest amount of unknown or out of vocabulary (OOV) words in a given application of a language, and it is often the case that transliteration is used to deal with them. Therefore, the transliteration mining process can be simplified, if we can identify and analyse only named entities. For the experiments in this section, we specified a regular expression for extracting words

[5]Retrieved from http://wikistats.falsikon.de/latest/wikipedia/en/

from each English Wikipedia article that started with an uppercase Latin character and was followed by lowercase character(s) irrespective of whether the word was linked or unlinked. The English named entities extracted in this way formed the set of candidate transliterations on the English side. It is important to note that depending on the named entity recognition (NER) requirements, the process of finding named entities from documents may not be as simple as specified here for our experiments, it may require the use of a sophisticated NER approach. For the Russian articles, we extracted only words that were written using Cyrillic characters (both lowercase and uppercase), also regardless of whether they were linked or unlinked text. From the set of English and Russian candidate named entities, we hand picked a subset of single word matches in the two languages to form the Gold standard set. Table 6.10 shows the article titles with the different number of words for the noisy data and gold set data. As table 6.10 shows, the total number of English words is less than the total number of Russian words since we also assumed that words starting with lower case characters were to be treated as candidate named entities on the Russian side[6].

English Wikipedia title	Total # words		gold standard size
	English	Russian	
The Beatles	822	1693	35
Ted Kennedy	1360	517	17
Michael Jackson	1505	2277	47
YouTube	694	1026	5
Perseids	55	330	2
District 9	575	1857	35
Hans Christian Ørsted	161	969	6
Inglourious Basterds	655	2271	41
Lady Gaga	852	1160	23
True Blood	880	732	61
Total of uniq words	4811	9334	264

Table 6.10: *Wikipedia article data for English and corresponding Russian articles.*

[6]The reason for including all lowercase Russian words is based on an identification of a topic pair where we found that the English side used uppercase for the word "atoll" in the topic "Johnston Atoll" whereas the Russian side had a lower case representation, i.e. "Джонстон (атолл)". We allow lowercase words in one language as candidate NEs to avoid a possible early loss of genuine transliteration pairs. However, we allow lowercase words in only one language and not both languages since we expect words that are not nouns between the languages to be handled easily by translation.

6.5.3 Evaluation setup and results

This task is similar to the first task with only one difference that here we analyse a higher number of candidate named entities from the article content than from the topic pairs. We therefore use the same evaluation measures of Precision, Recall, and F-score as expressed in Equations 6.1, 6.2, and 6.3 respectively. One plausible evaluation approach is to compute the measures for different cut-offs on the number of high-ranking pairs of words that are returned after applying a given transliteration mining system. Using these cut-offs should give us a good impression of the performance of a particular approach in ranking true transliteration pairs before non-transliteration pairs (Manning and Schütze 1999). We trained the Pair HMMs using the Baum-Welch EM algorithm as described previously and the transduction-based DBN models using a generalized expectation maximization algorithm as described in Chapter 5.

Figure 6.5 shows the precision-recall curves resulting from the different cut-offs after applying three Pair HMMs (PHMM3, PHMM5, and PHMM9) where the forward log-odds algorithm is used to compute transliteration similarity, and two context-dependent DBN models which model a dependency of the edit operation variable on the current source character (CONs1) and the the current target character (CONt1). We also plot the F-score values against the cut-offs associated with the different edit distance-based DBN models as shown in Figure 6.6.

As Figure 6.6 shows, only one edit distance-based DBN model achieves an F-score value higher than 0.7 which is unlike the case of the English-Russian transliteration mining results in the first task where a Pair HMM and a context-dependent DBN model result in F-score values greater than 0.8. In Figure 6.5, we see that at all recall values, PHMM3 has the lowest precision in comparison to the other Pair HMMs. PHMM5 results in the best performance of all Pair HMMs and even performs better than the context-dependent DBN models at higher levels of recall. The differences in the curves between the Pair HMMs in Figure 6.5 are greater than the differences in the Precision-Recall curves that we plotted from using Pair HMMs on transliteration data for English-Hindi and English-Tamil in the first task. It seems that for the second task in this section where we have a lot of noisy entities in evaluation data, the use of the forward log-odds algorithm leads to greater changes in transliteration mining quality with changes in Pair HMM transition parameters compared to the case for the transliteration detection task in Chapter 4.

The transliteration mining results in Figure 6.5 also suggest that context-dependent DBN models lead to a better discrimination between true transliteration pairs and non-transliteration pairs at lower levels of recall where we expect the models to rank true transliterations higher than non-transliterations. The results at lower levels of

recall for this second task concur with the results for the shared task setup in Table 6.5 where the use of a context-dependent model leads to an improved F-score value over the Pair HMM with distinct transition parameters (PHMM9F) where the forward algorithm is used to compute transliteration similarity. The curves in both

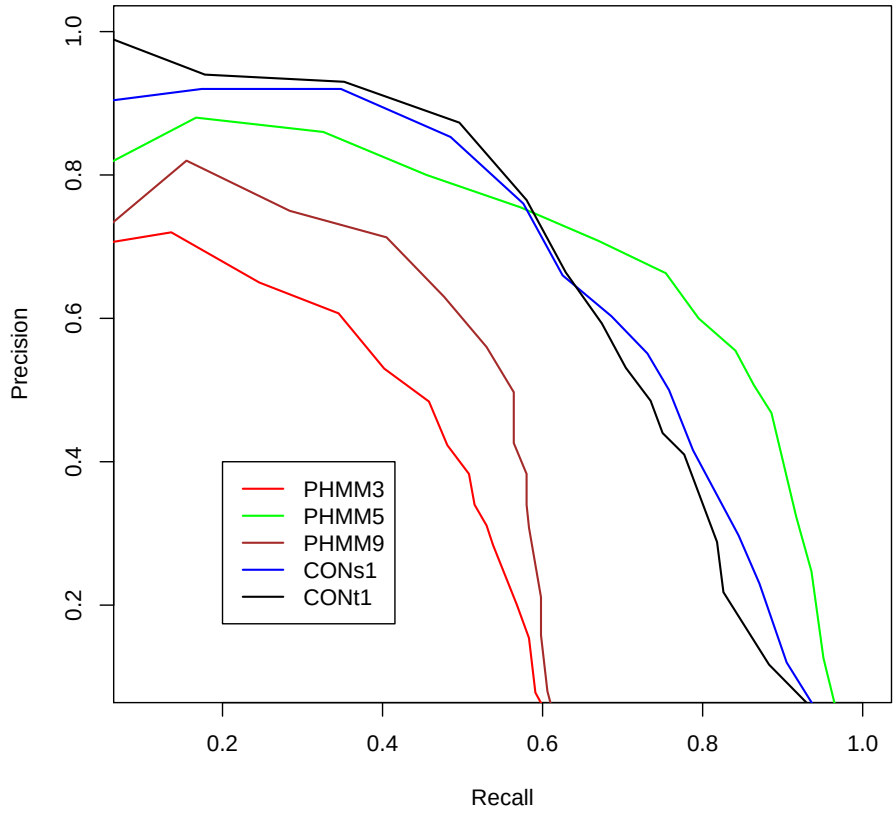

Figure 6.5: *Precision-Recall curves for Pair HMMs and context-dependent DBN models after evaluation on mining transliteration pairs from English-Russian comparable Wikipedia articles. CONs1 models the dependency of the edit operation variable on the current source character and CONt1 models the dependency of the edit operation variable on the current target character. PHMM3, PHMM5, and PHMM9 have different transition parameter settings with PHMM3 using three transition parameters, PHMM5 using five transition parameters, and PHMM9 using nine transition parameters.*

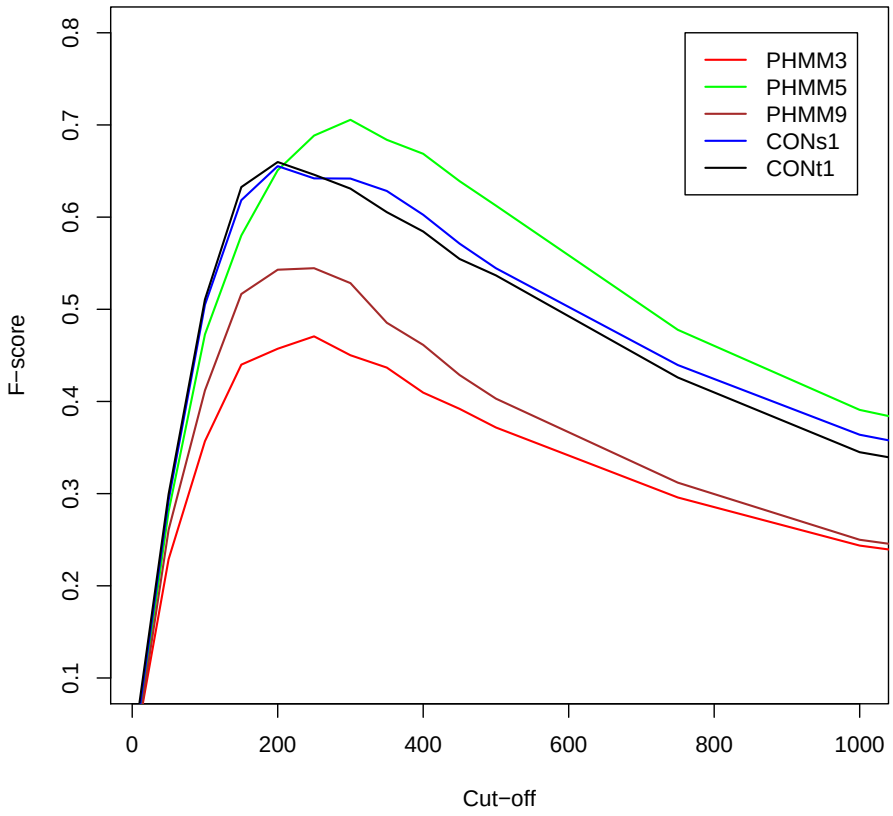

Figure 6.6: *Graph of F-score against returned cut-off for Pair HMMs and context-dependent DBN models after evaluation of mining transliteration pairs from English-Russian Wikipedia articles.*

Figures 6.5 and 6.6 suggest that for the edit distance-based DBN models to be valuable in mining transliteration pairs, we need to consider only a limited number of the high ranking candidate transliteration pairs returned by the system.

6.6 Conclusion

This chapter has showed the possibility of applying Pair HMMs and context-dependent DBN models in mining transliterations from real-world data (cross-language Wikipedia

data in our case). The results from the first task associated with mining transliterations from Web-based cross-language topics, show an excellent performance by the Pair HMMs on the English-Hindi and English-Tamil datasets. The Pair HMMs also resulted in relatively better transliteration mining quality than the context-dependent DBN models on the two datasets. A comparison with methods that performed well in mining transliteration pairs from the inter-language topics in a recent transliteration mining shared task (Kumaran et al. 2010b) showed comparable and in some cases better F-score values. For the English-Russian dataset, the use of a context-dependent DBN model resulted in a higher F-score value than a Pair HMM using distinct transition parameters.

However, we see that the quality of transliteration mining results for the second task, where we use additional content mainly from the body of each of the cross language Wikipedia articles, is not as good as for the first task. Nonetheless, the results show that the Pair HMMs and context-dependent DBN model can still be valuable if we consider up to an appropriate limit of the ranked list of potential transliteration pairs returned by the system. An implication here is that the edit distance-based DBN models could be useful for mining transliteration pairs not only from Wikipedia but also from other Web resources such as from bilingual or multilingual News websites which are frequently updated with new articles and where the occurrence of named entities is higher.

Chapter 7

Applying Pair HMMs in transliteration generation

7.1 Introduction

This chapter extends our investigation of using Dynamic Bayesian Networks (DBNs) to compute transliteration similarity for generating candidate transliterations. The transliteration detection and mining tasks in the previous three chapters required detecting transliteration pairs from a set of candidate named entities (NEs). The transliteration generation task in this chapter requires converting an input NE from a 'source' language to one or more representative 'target language' NEs. Although the transliteration detection and mining results in the previous three chapters show that various Pair Hidden Markov Models (Pair HMMs) and transduction-based DBN models lead to high transliteration detection and mining accuracy, in this chapter we aim to determine whether using Pair HMMs in generating candidate transliterations given source words and vice versa could result in a transliteration generation performance similar to that for the transliteration detection and mining tasks. The transliteration generation task is, however, more challenging than the transliteration detection or mining tasks, since without knowledge about candidate NEs, the search space for a representative target language candidate transliteration given a source language input NE is many times larger.

The transliteration generation process relies on two main stages, each of which can easily affect output quality. The first stage is concerned with the segmentation of

This chapter is an extended version of the following publications:
P. Nabende – Transliteration System using Pair HMMs and weighted FSTs, *Proceedings of the 2009 Named Entities Workshop*, pp. 100–103, Suntec, Singapore.
J. Tiedemann and P. Nabende – Translating Transliterations, *International Journal of Computing and ICT Research*, 03(1):34–43.

an input string and the second is concerned with the mapping of each segment to a correct representation in the target writing system. Although the latter stage can be addressed through a transliteration model such as the ones we are about to investigate, there is an increased level of ambiguity in mapping characters from one writing system to another in comparison to the transliteration detection or mining tasks. In the transliteration generation process, the number of choices that we can associate with character relationships, can in theory be approximated to the product of the size of the character vocabularies between the source and target language including transformations involving an empty symbol. It is also often the case that character transformations in writing systems could be one-to-many (for example, the Russian ⟨ч⟩ to the English ⟨ch⟩) and many-to-many. Recent work on transliteration generation does take this into account and alignment algorithms that induce different kinds of relationships have been developed (Jiampojamarn et al. 2010). In this chapter, our main interest is in the use of Pair HMM parameters for generating transliterations. Pair HMMs were initially designed for sequence alignment (Durbin et al. 1998), but their capacity to relate source and target elements using probabilistic values enables us to specify them as weighted finite state automata that we can use for transliteration generation. More specifically, we can represent the probabilistic values that relate pairs of source and target language elements in the edit operation states of a Pair HMM as transduction parameters. We can therefore represent the probabilistic values as Weighted Finite State Transducers (WFSTs) which can be used in decoding candidate target transliterations for a given input source word. In that regard, we would like to know whether the use of Pair HMM parameters when represented as WFSTs can lead to any gains in transliteration generation.

Traditionally, transliteration generation involves different writing systems, and we shall first evaluate the models on standard transliteration generation corpora (Li et al. 2009, Li et al. 2010) where each language in a pair uses a different writing system. Apart from the traditional view of transliteration generation, we also introduce a task that according to our knowledge has not yet been addressed in the transliteration generation literature. As we noted in section 4.4, the output of cross-lingual applications when encountering 'unknown' words between two languages with the same writing system is affected in the same way as between two languages with different writing systems. We also propose to use the traditional transliteration generation setup in dealing with 'unknown' words between languages that use the same writing system in addition to using the traditional transliteration detection setup. In section 7.2, we continue with the descriptions for the two tasks; in section 7.3.1, we introduce finite state automata concepts and show how to represent Pair HMMs as such; later we evaluate several weighted finite state automata including Pair HMM-based WFST models.

7.2 Transliteration generation tasks

7.2.1 Traditional machine transliteration task

The traditional transliteration generation task emphasizes the mapping of symbols from one writing system to symbols in another writing system. A survey of the language pairs involved in all three shared tasks on transliteration generation so far (Li et al. 2009, Li et al. 2010, Zhang et al. 2011) clearly shows that each language pair involves languages that use different writing systems. In many cases, automatic transliteration generation involves a language that uses the Latin alphabet. Again, a good case in point is with all the three shared tasks on transliteration generation where each language pair dataset consists of English data as source or target language data. The vast amount of research on automatic transliteration generation for which the Latin alphabet is involved may be attributed to not only the Latin alphabet's wide usage, but also to the simplicity that is associated with processing Latin characters. However, with recent advances on the part of the digital encoding of symbols from various writing systems (such as with the UTF8 encoding system), there is now some research on automatic transliteration generation between different writing systems that do not involve the Latin alphabet (Malik 2006, Malik et al. 2008). For this task, we shall experiment with standard transliteration corpora from the NEWS 2009 and NEWS 2010 shared tasks on transliteration generation (Li et al. 2009, Li et al. 2010).

7.2.2 Translating transliterations task

In this section, we argue that the transliteration generation task need not be restricted to the case where the source and target language use different writing systems. This is mainly because the limitations of NLP systems for which transliteration is used across writing systems are similar to those involving different languages that use the same writing system. Specifically, 'unknown' entities will affect a cross-lingual application that involves languages that use the same writing system in a similar way that they affect a cross-lingual application that involves different writing systems. However, to the best of our knowledge, the handling of 'unknown' entities in a cross-lingual application where the source and target language use the same writing system in the context of using the traditional transliteration generation setup has yet to be addressed in literature. In this chapter, we are mainly concerned with the case where transliterated names originating from a different writing system differ across languages using the same writing system. Consider the examples in Table 7.1 where Russian names have been transliterated into: English, French, German, and Dutch. As can be seen, the transliterated names are spelt differently although the four languages use the same alphabet. Such spelling variations given the same

English	French	German	Dutch
Alexander Pushkin	Alexandre Pouchkine	Alexander Puschkin	Aleksandr Poesjkin
Nikita Khrushchev	Nikita Chruschtschow	Nikita Khrouchtchev	Nikita Chroesjtsjov
Yuri Andropov	Iouri Andropov	Juri Andropow	Joeri Andropov
Leonid Brezhnev	L'eonid Brejnev	Leonid Breschnew	Leonid Brezjnev

Table 7.1: *Transliterated Russian names in four European languages that use the Latin alphabet.*

original name arise due to language specific differences.

The problems addressed in the traditional machine transliteration framework should apply here as well. The different spelling variants try to match the underlying phonetic description for the original word and are very likely to be unknown by a cross-language application regardless of whether the languages involved use the same writing system or not. We expect a dedicated module for transliterating the 'unknown' transliterated words to improve the performance of a cross-language processing system between languages using the same writing system in the same way we expect a transliteration module to improve system performance across writing systems.

7.3 Using Pair HMMs in transliteration generation

Hidden Markov Models (HMMs) have been applied before in statistical machine translation to align words (Vogel et al. 1996). In a manner similar to that of applying classic HMMs to word alignment, we use Pair HMMs to estimate parameters for relating source and target language characters which we use for suggesting candidate transliterations given a source language word. A transliteration generation module is obviously needed to help use the Pair HMM parameters as transliteration generation parameters. One natural option is to develop our own decoding module using Pair HMM parameters. This module would be required to facilitate the representation of the edit states of a Pair HMM and the emission parameters encoded there. Although Pair HMM emission parameters (Section 4.3) relate source and target string elements and may be sufficient for modeling transliteration generation, the transliteration detection results in Chapter 4 suggested the importance of using Pair HMM transition parameters for computing transliteration similarity. Therefore, it is important that the decoding module also facilitates incorporating Pair HMM transition parameters to compute transliteration similarity for suggesting candidate transliterations.

Instead of developing our own transliteration generation module, we follow a plau-

sible approach of using Pair HMMs as finite state automata. We have already applied Pair HMMs in a manner similar to how a finite state automaton (*acceptor*) would be used to accept or reject input. Since a Pair HMM relates elements from source and target vocabularies, we can use them (Pair HMMs) for generating transliterations if they are represented as finite state *transducers*. Pair HMMs find their origins as word alignment finite state automata, and the reverse process of representing them (Pair HMMs) as finite *transducers* should not be difficult to achieve. One advantage of following the approach of representing Pair HMMs as automata is that there already exist various software tools that enable the implementation of different types of finite state automata and therefore our only task would be to specify how a given software tool would implement an automaton that corresponds to a given Pair HMM. The requirements for transliteration generation that have been noted in the previous paragraph for a transliteration generation module apply here as well. In the next subsection we review the concepts associated with finite state automata, and later we describe how we have represented Pair HMMs as finite state transducers.

7.3.1 Finite state automata

A finite state automaton is defined as a mathematical abstraction which refers to a model consisting of a finite number of states, a set of transitions between the states, and actions (Jurafsky and Martin 2009). Each transition is described by a condition that needs to be fulfilled so as to enable a state change, and an action describes an activity that is to be performed at a given moment.

Finite state automata are broadly categorized into two: finite state acceptors (FSAs) and finite state transducers (FSTs). The difference between the two is associated with the representation on the transition arcs of the the automaton: an FSA is used to only accept or reject input elements and an FST is used to map from an input element to an output element. Hence, an FSA defines a model only for input elements on the arcs and an FST defines a model that relates input and output elements. Automata from each of these categories can be applied in a transliteration generation framework. Of course, we can apply a finite state transducer to generate a target string from some source string. And we can apply a finite state acceptor to check whether the target string that was generated conforms to the spelling (or pronunciation) regularities of the target language.

It is often convenient to represent finite state automata using state transition diagrams where nodes denote states and edges are labeled with symbols. Figure 7.1 is a state transition diagram illustrating a finite state acceptor that will accept any combination of the symbols ‘a’ and ‘b’ from a two symbol alphabet {a,b} if and only if the first symbol is an ‘a’. In order to use a finite state acceptor, the following

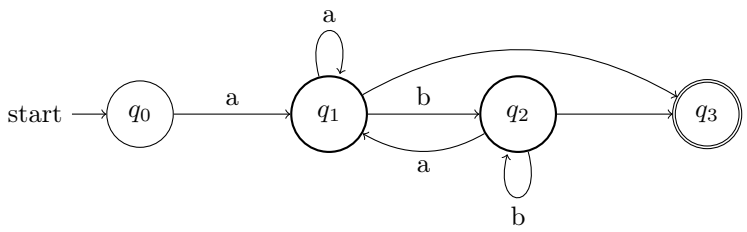

Figure 7.1: *An example of a finite state acceptor that will accept only strings that start with the symbol 'a' using a two symbol alphabet {a,b}. Following usual convention, the start state q_0 is represented with an incoming arrow and the final or accepting state is represented by the double circle.*

parameters need to be defined (Jurafsky and Martin 2009):

- a finite set of states $Q = q_0, q_1, ..., q_n$
- a finite set corresponding to the input alphabet Σ
- the start state q_0
- the set of final states, $F \subseteq Q$
- the transition function $\delta(q, i)$ or transition matrix between states. The transition function defines a mapping to a new state $q' \in Q$ given a state $q \in Q$ and an input symbol $i \in \Sigma$.

The transliteration generation task mainly requires the use of finite state transducers. A finite state transducer defines a relation between sets of strings and therefore enables a mapping from one representation to another. Jurafsky and Martin (2009) summarize four ways of perceiving an FST as follows:

- An FST as a recognizer. In this case an FST takes a pair of strings as input and outputs; and accepts the strings if the string-pair is in the string-pair language, and rejects if it is not. This view of a transducer can be adapted and applied to the transliteration identification and mining tasks in the previous three chapters.
- An FST as a generator. In this case the FST outputs pairs of strings of the language with a corresponding 'yes' or 'no'.
- An FST as a set relator. In this case, an FST is used to compute relations between sets.

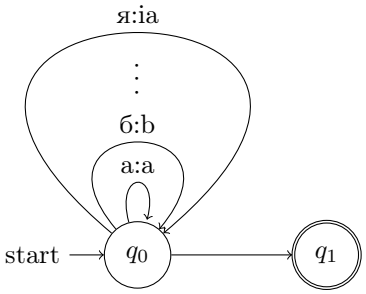

Figure 7.2: *An FST for a standard Cyrillic romanization system.*

- An FST as a translator. In this case the FST reads a string and outputs another string. This view of an FST is the most suitable for fulfilling the transliteration generation requirements and it is the one we shall use for the task.

Of the four metaphors, the 'FST as translator' metaphor relates more to using an FST as a transliterator. In the most basic case, a standard transliteration system can be represented as an FST. The incomplete Figure 7.2 illustrates a two state transducer where the start state is used to map characters from the Cyrillic alphabet to the Latin alphabet using the *post-2010 Passport system*[1] for romanizing Russian. In Figure 7.2, the arcs are labeled with input-output elements seperated by a colon.

In order to use a finite state transducer, the following parameters need to be defined (Jurafsky and Martin 2009):

- a finite set of states $Q = q_0, q_1, ..., q_n$
- a finite set corresponding to the input alphabet Σ
- a finite set corresponding to the output alphabet Δ
- the start state $q_0 \in Q$
- the set of final states $F \subseteq Q$
- the transition function $\delta(q, w)$ between states. $\delta(q, w)$ returns a set of new states $Q' \in Q$ given the current state q and an input string $w \in \Sigma^*$.
- the output function $\sigma(q, w)$ for determining the set of all possible output strings for each state. $\sigma(q, w)$ gives a set of output strings $o' \in \Delta^*$ given the current state $q \in Q$ and an input string $w \in \Sigma^*$.

[1] `http://en.wikipedia.org/wiki/Romanization_of_Russian#New_system_2010`

Finite state automata of the type in Figure 7.1 and 7.2 can only be useful for a limited number of applications. In the transliteration generation process where we encounter a lot of ambiguity in mapping symbols from one writing system to another, there is need to use a probabilistic approach. A natural approach that is commonly used to achieve probabilistic modeling involves the augmentation of a finite state automaton such that each arc is associated with a probability representing the likelihood of taking a given path, and that the probability of all arcs leaving a given state sums to one. An FSA (respectively FST) that associates each arc with a probability is referred to as a *weighted finite state acceptor* (WFSA) (respectively *weighted finite state transducer* (WFST)). WFSAs and WFSTs are formally defined as tuples over a semiring $\mathcal{K}$.

A WFSA is defined as a 7-tuple $\langle \Sigma, Q, q_0, F, E, \lambda, \rho \rangle$ where Σ, Q, q_0, and F are as defined above. $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times K \times Q$, refers to the set of transitions; $\lambda \in K$, refers to the *initial weight*; and $\rho : F \mapsto K$, refers to the final *weight function.*

A WFST is a 8-tuple $\langle \Sigma, \Delta, Q, q_0, F, E, \lambda, \rho \rangle$ where Σ, Δ, Q, q_0 and F are as defined above. $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times K \times Q$, refers to the set of transitions; $\lambda \in K$, refers to the initial weight; and $\rho : F \mapsto K$, refers to the *final weight function* mapping final states to elements in $\mathcal{K}$.

In Chapter 2, we reviewed one of the earliest application of weighted finite state automata in transliteration generation, that is for Japanese Katakana to English back-transliteration (Knight and Graehl 1997). In some of the experiments we will test various weighted finite state automata while applying them in a manner similar to how they have been previously applied to generate transliterations. However, a major aim in this chapter is to determine the value associated with representing Pair HMMs as WFSTs and using the resulting Pair HMM-based WFSTs for transliteration generation.

7.3.2 Representing Pair HMMs as WFSTs

The emission states of a Pair HMM encode (like in a transducer) the relationship between source and target language elements. In order to use the Pair HMM parameters as transduction parameters, we first specify an FST structure which approximates that of a Pair HMM; and later specify the integration of Pair HMM emission and transition parameters on the arcs of the FST structure. Figure 7.3 shows a finite state automaton that approximates the Pair HMM with distinct emission and transition parameters (see Figure 4.4). In Figure 7.3, the Pair HMM emission parameters for a particular edit state are represented on the transition arcs that are directed towards a similar state in the automaton. The transition parameters between Pair HMM states are represented on corresponding transition arcs between similar states of the

FST. Therefore, the probability that relates source and target elements in the FST of Figure 7.3 is the product of the Pair HMM transition probability from the previous state to the state (where the relationship is modeled) and the Pair HMM emission probability associated with the pair of symbols. For example, assuming that p_{x_i,y_j} is an emission probability which relates the source element (x_i) and the target element (y_j) in the Pair HMM substitution state (M); the probability associated with relating these two symbols for a transition from the deletion state (X) to the match state (M) in the corresponding FST is specified as $(1 - \epsilon_X - \lambda_X - \tau_X) \times p_{x_i,y_j}$. In Figure 7.3,

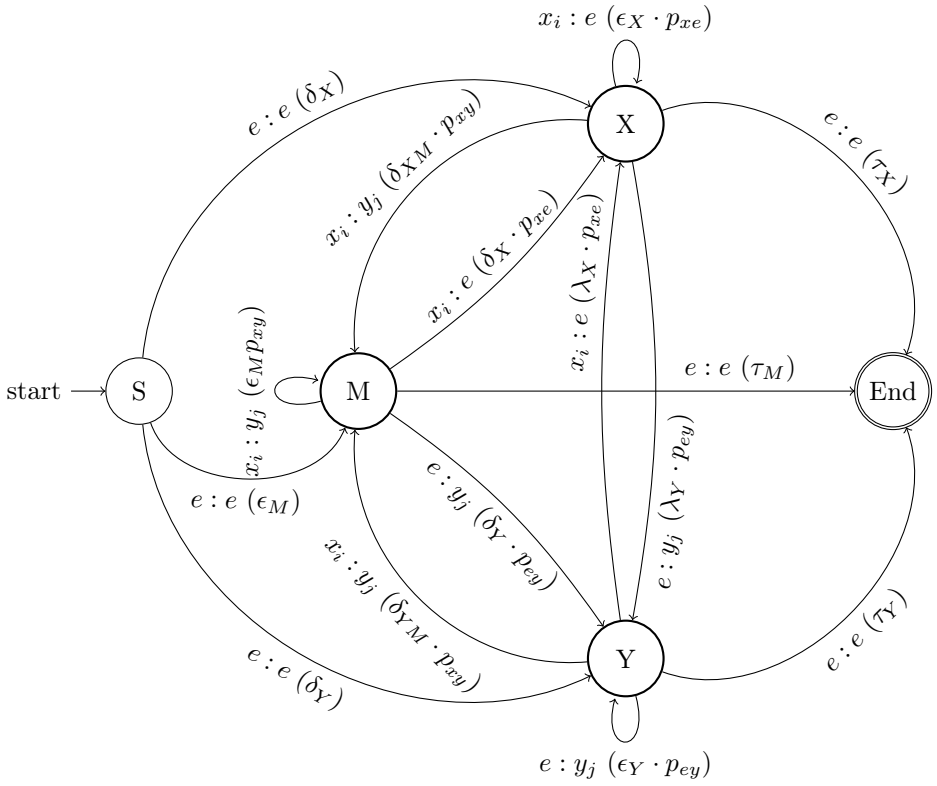

Figure 7.3: *A finite state transducer approximation of the Pair HMM with nine transition parameters. The input and output elements separated by a colon are shown just before the parentheses on each arc of the transducer.* e *is used to represent the empty symbol. The combination of Pair HMM transition and emission parameters are as shown on each arc. The emission probabilities ($p_{..}$) correspond to the emission probabilities in the Pair HMM edit states that model the relationship between the input and output elements.*

we also define a start state to explicitly capture the starting parameters for the Pair HMM. In the transliteration detection and mining tasks, we assumed the Pair HMM to start in any of the three edit operation states using the transition parameters from the substitution state to each of the respective three edit states. We assume the same setup of starting parameters for the transliteration generation task.

7.4 Experiments using NEWS 2009-2010 shared task data

7.4.1 Data

The NEWS 2009-2011 shared tasks on transliteration generation datasets involved different writing systems between language pairs, with English being either a source or target language for each language pair. As shown in Table 7.2 we use seven of the 12 language pairs that were provided: English→Bengali, English→Hindi, English→Kannada, English→Russian, English→Tamil, English→Thai, and Thai→English.[2]

Language pair	Training	Development	Testing
English→Bengali (En-Be)	13000	1000	1000
English→Hindi (En-Hi)	10014	2099	1000
English→Kannada (En-Ka)	8065	2108	1000
English→Russian (En-Ru)	5977	943	1000
English→Tamil (En-Ta)	8037	2184	1000
English→Thai (En-Th)	27668	1948	2000
Thai→English (Th-En)	24051	1793	1994

Table 7.2: *Size of training, development, and testing datasets per language pair. Source: NEWS 2009 and NEWS 2010 transliteration generation shared task data (Li et al. 2009, Li et al. 2010)*

a) Data pre-processing

To help reduce on data sparseness, we ensured only lowercase representation for languages (English and Russian) where conversion to only lowercase was necessary. For

[2]The other language pairs from the two shared tasks include: English→Chinese, Chinese→English, English→Japanese Katakana, English→Japanese Kanji, and English→Korean Hangul. Apart from the English→Chinese and Chinese→English datasets, we did not use the other three datasets because they were expensive to purchase.

some language pairs, some source language words had at least two transliteration variants in the training dataset. During training each variant was matched to the source word and used individually as a training pair. Some data sets also contained some multi-word sequences. We followed an approach similar to the one used by Finch and Sumita (2010) in handling multi-word sequences. For those multi-word sequences where the number of words in the source and target word sequences matched, we split the word sequences into individual words; during training, we matched words in the same word position in the source and target word sequence and used the resulting individual word pairs as training data; during testing, each source word from the multi-word sequence was transliterated individually, and the *n-best* transliteration lists associated with all the individual words in the source word sequence were subsequently combined into a single output transliteration sequence. For the other multi-word sequences in the training data where the numbers of words in the source and target word sequences differed, we introduced a ⟨space⟩ token into the sequence, and treated it as one long sequence.[3]

7.4.2 Transliteration models

a) WFST Parameter estimation

We apply two sets of WFSTs to transliteration generation. First, we apply WFSTs in the usual way. The second set of WFSTs are approximate representations of Pair HMMs. Both sets of WFSTs use the notion of edit distance.

We varied the first set of WFSTs in terms of the states and their possible emissions. For one of these WFSTs, we use a structure corresponding to that of a Pair HMM (Figure 7.4). As Figure 7.4 shows, this WFST also uses separate states for substitutions, insertions and deletions. In this type of WFST, we introduce some kind of model bias by restricting the type of emissions to be of a certain kind at each state. In a different setting, we removed this bias by allowing all possible types of emissions (including insertions on the source and target side) from any state of the WFST. The idea in this case is to let the training procedure decide how to make use of the hidden layer of states without defining the function of each state. This is basically a test to see if the forward-backward parameter estimation procedure (which is used for training the WFSTs) is capable of learning some underlying structure which is not given to the system when training its parameters. However, we still have to define the number of states to be used in the WFST before training its parameters. In our experiments, we applied WFSTs with one to five states (excluding start and

[3]The other option that we could have followed for multi-word sequences where the number of words differed would have been to just ignore them during training as is the case in Jiampojamarn et al.'s work (2010).

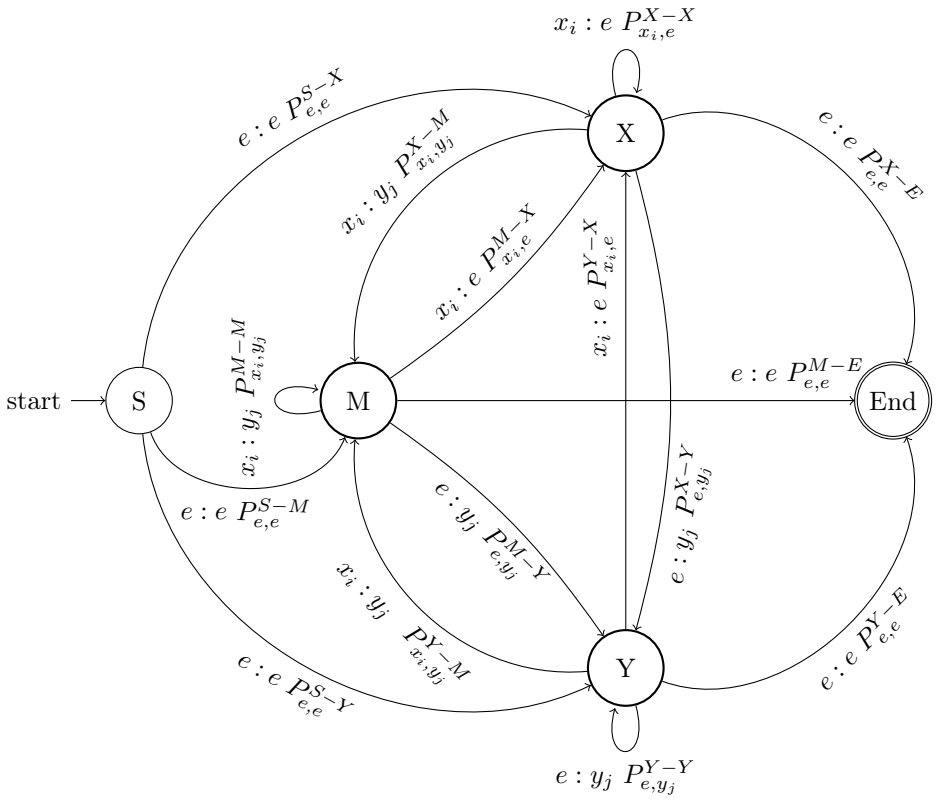

Figure 7.4: *An edit distance-based WFST.* x_i *and* y_j *represent elements from the source and target vocabulary.* e *is an empty symbol and* $P^{q-q'}_{\cdot,\cdot}$ *represents the probability of a relationship between the source and target elements from state* q *to state* q'.

end state) and a fully connected graph with uniform initial settings. Furthermore, we also ran the training procedure with three additional randomly chosen initial parameters. For this set of WFST models, we used a publically available finite state automata toolkit called CARMEL[4] for parameter estimation.

For the second set of WFST models, we first estimate Pair HMM parameters as described in Chapter 3; the Pair HMM parameters are then transformed into WFST transliteration generation parameters as described above and represented in a format suitable for use by the CARMEL software toolkit.

[4] `http://www.isi.edu/licensed-sw/carmel`

b) Phrase-based statistical machine translation

In addition to the WFSTs, we also tested the use of a phrase-based statistical machine translation approach on the English→Russian language pair. Phrase-based statistical machine translation (PSMT) is the current state of the art in data-driven machine translation, and has recently been applied to transliteration generation (Matthews 2007, Finch and Sumita 2008). It is based on the well-known IBM models which are trained on large parallel corpora but use bilingual phrase tables instead of word link probabilities and fertility parameters. In the PSMT approach, various components are usually combined in a log-linear model (translation models, reverse translation model, word and phrase penalties, language models, distortion parameters, etc) with weights optimized using minimum error rate training (MERT). Various tools are available for training such a model and "decoding" (translating) input strings according to the model. In our case, we used the publicly available toolkit Moses (Koehn et al. 2007) with its connected tools: GIZA++ (Och and Ney 2003) and IRSTLM (Federico et al. 2008). As a requirement of the transliteration generation task, we split the names on the character level. Machine translation applications usually require output word / phrase re-ordering. But for character-based translation / transliteration, we expect a monotonic ordering in the output that corresponds to the input. We therefore ensured that the PSMT system uses monotonic decoding. We left the other parameters for the Moses PSMT decoder unchanged. The model therefore uses the standard settings for character alignment with GIZA++, standard heuristics for the extraction and scoring of phrase alignments (character n-grams with a maximum length of 7 characters) and standard settings for the minimum error rate training (MERT) when tuning the models. The language model for the English-Russian case is a 5-gram model which we estimated from the target language side of the training dataset using the Witten-Bell smoothing technique which is implemented in the IRSTLM toolkit. There are various fixed parameters that can be tuned in the PSMT models. Among others, we could change the maximum size of phrases to be considered, various phrase extraction techniques can be used and language model parameters can be modified. In our setup, we did not tune these training specific parameters.

A major advantage of the PSMT approach over the weighted finite state transducers described above is that the extracted phrase tables (character n-grams) cover a lot of contextual dependencies found in the data. By exploiting these, we hope to find better transformations by translating sequences of characters instead of single characters. Furthermore, we do not have to model insertions and deletions explicitly but leave it to the translation table to change the lengths of translated strings. Another advantage is the explicit inclusion of a target language model to weight the

possible outcomes of the system. In the transducer model, this is not easily possible as we include deletion operations. The reason being that the language model would always prefer shorter strings and therefore force the system to over-use the deletion operations when transforming strings. Of course, we do not expect the WFSTs to perform better than the PSMT approach in the transliteration generation tasks. The use of a PSMT system, however, gives us an idea of the extent to which transliteration generation quality is affected by the limitations of the WFSTs and whether trying to address some of these limitations could lead to improved quality.

7.4.3 Evaluation metrics

We follow the same evaluation setup as used for the NEWS 2009 and NEWS 2010 shared tasks on transliteration generation. For each source language word in the test set, a participating system was required to generate and submit 10 best candidate transliterations. For cases where a source language word had alternative transliterations, all the alternatives were treated equally in the evaluation process. In the NEWS 2009 shared task on transliteration generation, six measures were used to evaluate transliteration generation quality. These include (Li et al. 2009): accuracy, fuzziness in Top 1 (mean F-score), mean reciprocal rank (MRR), mean average precision for reference transliterations (MAP_{Ref}), mean average precision in 10 best candidate transliterations (MAP_{10}), mean average precision for the system (MAP_{sys}). We use the same notation in Li et al. (2009) to define the evaluation metrics:

- N: total number of names (source words) in the test set
- n_i: number of reference transliterations for i^{th} name in the test set ($n_i \geq 1$)
- $r_{i,j}$: j^{th} reference transliteration for i^{th} name in the test set ($1 \leq j \leq n_i$)
- K_i: number of candidate transliterations produced by a transliteration system
- $c_{i,k}$: k^{th} candidate transliteration (output by the transliteration system) for the i^{th} name in the test set ($k \leq K_i$).

a) Word accuracy in Top-1 (ACC)

This measures the correctness of the first transliteration candidate in the n-best candidate list produced by a transliteration system. ACC = 1 means that all top candidates are correct transliterations, that is, they match one of the references, and ACC = 0 means that none of the top candidates are correct.

$$\text{ACC} = \frac{1}{\text{N}} \sum_{i=1}^{\text{N}} \{1 \text{ if } \exists r_{i,j} : r_{i,j} = c_{i,1};\ 0 \text{ otherwise}\} \tag{7.1}$$

Sometimes, the related metric of word error rate (WER) is used instead.

$$\text{WER} = 1 - \text{ACC} \tag{7.2}$$

b) Fuzziness in Top-1 (mean F-score)

The mean F-score measures how different, on average, the top transliteration is from its closest reference. F-score for each source word is a function of Precision and Recall and equals 1 when the top candidate matches one of the references and 0 when there are no common characters between the candidate and any of the references. Precision and Recall are calculated based on the length of the longest common subsequence (LCS) between a candidate transliteration and the reference transliteration:

$$\text{LCS}(c, r) = \frac{1}{2}(\text{length}(c) + \text{length}(r) - \text{ED}(c, r)) \tag{7.3}$$

where $\text{ED}(c, r)$ is the edit distance. For example, the longest common subsequence between "abcd" and "afcde" is "acd" and its length is 3. The best matching reference, that is, the reference for which the edit distance has the minimum is used. If the best matching reference $r_{i,m}$ is given as $r_{i,m} = \underset{i}{\text{argmin}}(\text{ED}(c_{i,1}, r_{i,j}))$, then Recall ($\text{R}_i$) and Precision ($\text{P}_i$) are calculated as follows:

$$\text{R}_i = \frac{\text{LCS}(c_{i,1}, r_{i,m})}{\text{length}(r_{i,m})} \qquad \text{P}_i = \frac{\text{LCS}(c_{i,1}, r_{i,m})}{\text{length}(c_{i,1})}$$

The F-score is computed as the harmonic mean of Precision and Recall (see Equation 6.3), sometimes referred to as F_1 score.

c) Mean reciprocal rank

Is obtained by averaging the reciprocal ranks of true transliterations. MRR closer to 1 implies that many correct transliterations are produced close to the top of the n-best lists. If a candidate that matches one of the references is in the j^{th} position in the n-best list, its rank is j and its reciprocal rank is $1/j$. Here, we would like the value for j to be as minimal as possible. If none of the suggested transliterations matches any of the reference transliterations, we assume a reciprocal rank of 0 based on the non-matching suggested transliterations for a particular source name in the test set.

$$\text{MRR} = \frac{1}{\text{N}} \sum_{i=1}^{\text{N}} \left\{ \min_j \frac{1}{j} \text{ if } \exists r_{i,j}, c_{i,k} : r_{i,j} = c_{i,k}; 0 \text{ otherwise} \right\} \tag{7.4}$$

d) Mean Average Precision$_{\text{reference}}$ (MAP$_{\text{reference}}$)

Measures the precision in the n-best candidates for the i^{th} source name, for which n_i reference transliterations are available. If all the references are produced, then MAP is 1.

$$\text{MAP}_{ref} = \frac{1}{\text{N}} \sum_{i=1}^{\text{N}} \frac{1}{n_i} \left(\sum_{k=1}^{n_i} \frac{\text{number of correct candidates for } i^{th} \text{ word in } k\text{-best}}{k} \right)$$

e) Mean Average Precision$_{10}$ (MAP$_{10}$)

MAP$_{10}$ measures the precision in the 10 best candidates for the i^{th} source name suggested by the transliteration generation system. In general, the higher MAP$_{10}$ is, the better is the quality of the transliteration system in capturing multiple references. The number of reference transliterations may be more or less than 10. If the number of reference transliterations is less than 10, then MAP$_{10}$ can never be equal to 1. Only if the number of reference transliterations for every source word is at least 10, then MAP$_{10}$ could possibly be equal to 1.

$$\text{MAP}_{10} = \frac{1}{\text{N}} \sum_{i=1}^{\text{N}} \frac{1}{10} \left(\sum_{k=1}^{10} \frac{\text{number of correct candidates for } i^{th} \text{ word in } k\text{-best}}{k} \right)$$

f) Mean Average Precision$_{\text{system}}$ (MAP$_{\text{sys}}$)

MAP$_{\text{sys}}$ measures the precision of the top K_i-best candidates produced by the system for the i^{th} source name, for which n_i reference transliterations are available. This measure allows the systems to produce a variable number of transliterations, based on their confidence in suggesting correct transliterations. If all the n_i references are produced in the top-n_i candidates (that is $K_i = n_i$, and all of them are correct), then MAP is 1.

$$\text{MAP}_{\text{sys}} = \frac{1}{\text{N}} \sum_{i=1}^{\text{N}} \frac{1}{K_i} \left(\sum_{k=1}^{K_i} \frac{\text{number of correct candidates for } i^{th} \text{ word in } k\text{-best}}{k} \right)$$

7.4.4 Results

a) English→Bengali, English→Hindi and English→Kannada results

For English→Bengali, English→Hindi and English→Kannada, we applied three Pair HMM-based WFST models as shown in Table 7.3. phmm0wfst refers to the WFST that captures only the emission parameters of a Pair HMM. We applied phmm0wfst with the aim of determining whether using only Pair HMM emission parameters

would suffice for the transliteration generation task. phmm5wfst refers to the WFST model that captures the parameters of a Pair HMM that uses five transition parameters (See Figure 4.2). phmm9wfst refers to the WFST model that captures the parameters of a Pair HMM that uses distinct transition parameters (See Figure 4.4).

Table 7.3 shows the results for three language pairs associated with the use of Pair HMM parameters in *standard runs* for the respective datasets. A *standard run* in this case refers to the use of only the training data that was provided for the NEWS 2010 shared task on transliteration generation. There is also a *non-standard* run, where participants can use additional external datasets. Table 7.3 results suggest that Pair HMM parameters based on one-to-one character alignments are not good at all for transliteration generation. None of the Pair HMM-based WFST models achieve accuracies comparable to those from the systems that participated in the NEWS 2010 shared task on the datasets for the three language pairs. A review of the participating systems in the NEWS 2010 shared task puts the 'basic' application of Pair HMM parameters at a clear disadvantage. All the participating systems including a phrase-based statistical machine translation (PSMT) approach (Finch and Sumita 2010), n-gram and joint source channel-based models (Das et al. 2010), and an online sequence prediction model based on many-to-many alignments modeled far more information than that represented in Pair HMM parameters. All models used by the shared task systems represented some character context information which was not represented in the Pair HMM parameters. However, the LCS-based F-score values suggest that the Pair HMM-based WFST approach seems to result in

Language pair	model	accuracy	F-score	MRR	MAP_{ref}
English→Bengali	phmm0wfst	0.021	0.641	0.035	0.021
	phmm5wfst	0.100	0.713	0.135	0.100
	phmm9wfst	0.100	0.713	0.132	0.100
English→Hindi	phmm0wfst	0.009	0.614	0.017	0.009
	phmm5wfst	0.030	0.654	0.052	0.030
	phmm9wfst	0.030	0.654	0.050	0.030
English→Kannada	phmm0wfst	0.010	0.621	0.012	0.010
	phmm5wfst	0.015	0.614	0.021	0.015
	phmm9wfst	0.015	0.614	0.030	0.015

Table 7.3: *Standard run transliteration generation results for three language Pairs. phmm0wfst is the WFST that uses parameters from a Pair HMM with no transition parameters between edit states; phmm5wfst is a WFST that uses parameters from a Pair HMM with five transition parameters (Figure 4.2); and phmm9wfst uses parameters from a Pair HMM with distinct emission and transition parameters (Figure 4.4).*

many correct character conversions. We can see that even for the worst performance on accuracy, none of the Pair HMM - based models had below 60% F-score. The table also shows that the use of Pair HMM transition parameters between edit states leads to an improvement in transliteration generation quality over the case when they are not used.

b) English→Russian transliteration generation results

For English→Russian transliteration generation, we also present results for models that participated in the NEWS 2009 shared task on transliteration generation (Li et al. 2009). Table 7.4 shows results from the application of different WFST models and the PSMT approach. The results for the models that participated are marked with an asterisk. The models with the extension _rules refer to the case where we modeled for English vowel bi-gram combinations and bi-grams associated with Cyrillic romanization, and a post-processing step that involved the use of a few transformation rules. When using development data, a check on the transliterations that were generated using Pair HMM parameters when applied in the most 'basic' way showed consistent mistransliterations. For example, in all cases where the Russian

Model	accuracy	F-score	MRR	MAP_{ref}
	Standard runs			
phmm0wfst	0.055	0.758	0.069	0.055
phmm9wfst	0.298	0.856	0.346	0.298
phmm5wfst*	0.293	0.845	0.325	0.293
phmm5wfst_rules*	0.354	0.869	0.394	0.354
Moses_PSMT*	0.509	0.908	0.619	0.509
	Non-standard runs			
phmm5wfst*	0.341	0.776	0.368	0.341
phmm5wfst_rules*	0.515	0.821	0.571	0.515
edit_WFST*	0.321	0.768	0.403	0.321
edit_WFST_rules*	0.466	0.808	0.525	0.466
Moses_PSMT*	0.612	0.845	0.660	0.612

Table 7.4: *Results for English→Russian transliteration generation. Models marked with an asterisk participated in the NEWS 2009 shared task on transliteration generation. phmm0wfst, phmm5wfst, and phmm9wfst are as defined in the caption of the previous Table. edit_WFST refers to the WFST that uses separate states for the different edit operations as shown in Figure 7.4. the extension _rules indicates the additional use of transformation rules in a post-processing step.*

character л 'l' precedes the Russian soft sign ъ '", the Russian soft sign was missing. For example крефелд and билбао were generated instead of крефелъд 'krefeld' and билъбао 'bilbao' respectively. This affected transliteration generation quality. For such cases, simple transformation rules such as "л→лъ" were used on the generated transliterations in a post processing step. 25 transformation rules were specified to help deal with some of the mistransliterations. The Moses_PSMT system was used with the settings described in subsection c.) above on 'Phrase-based statistical machine translation'. Although the performance of the Pair HMMs is lower than that for the PSMT system, the results in Table 7.4 show that the use of contextual information in the Pair HMMs and the application of contextual rules in a post-processing stage improves transliteration generation quality. In both cases the F-score for Pair HMMs that use the additional information approaches that of the PSMT system. These results also prove that Pair HMM transition parameters are important for transliteration generation. For the *non-standard* runs we used additional English-Russian data from the geonames database (Chapter 3) to train the models. As results show, the use of additional data leads to a general improvement in transliteration generation quality for all models, and the Pair HMM-based method where some context is used with post-processing transformation rules results in a relatively greater improvement.

7.5 Experiments on translating transliterations

In this set of experiments, we investigate the application of transliteration models similar to those that we applied in the previous section. Here we evaluate the models on translating transliterations between English and Dutch, and between English and French. For some models, we investigate further settings in addition to those that have been used in the previous subsection.

7.5.1 Data

We extracted the datasets for this set of experiments from an English Wikipedia data dump from 2008/07/24 in the same way we extracted corresponding named entities for training edit distance-based DBN models in the second transliteration mining task in Chapter 5. We also used simple patterns to identify Russian names looking at the structured information in Wikipedia info-boxes. We looked at entries that match the pattern (Russian|Russia|Soviet) in categories such as "citizenship", "nationality" and "place of birth". Translations of these names are taken from Wikipedia inter-language links (WILs) which exist on every source page. We collected all names potentially from Russian origin and their correspondences in other languages. We saved all name

pairs for the language pairs we were interested in, performing some extra normalization similar to that described in Chapter 5. This includes: normalizing of names that had abbreviations (e.g. "George H.W. Bush") and / or a middle name (e.g. "Nikita Sergejewitsch Chruschtschow"); and switching the order of first and family names (e.g. "Clinton, William Jefferson" instead of "William Jefferson Clinton"). These pre-processing steps, however, resulted in a small dataset for each language pair. We obtained 199 pairs of names for English-Dutch and 372 pairs for English-French. We did not manually check them and, therefore, this data includes names which are not typically Russian (such as Marc Chagall, born in the Russian empire as a son of a Jewish family). However, we assume that there are only very few of these exceptions. From each of our datasets, we designated 50 name pairs for testing. Each test set is used for evaluating all the models described in the next subsection. The remaining pairs were used for training and / or tuning model parameters.

7.5.2 Transliteration models

a) WFST parameter estimation

The WFSTs in this set of experiments are trained just as described in the previous task. For this set of experiments, we ran the training procedure with a uniform initial model and five other randomly chosen initial models which are aimed at reducing the likelihood of ending up in a suboptimal model. We first applied the edit distance model (see Figure 7.3) which implements separate states for substitutions, insertions, and deletions. Then, we also applied various FSTs where we varied the number of states while letting the training procedure decide on how to use the hidden layer of states. We also modified the source and target language alphabets by changing the way of splitting strings into symbol sequences. Previously, we simply used character sequences for training and testing. For this task, we split the words into sequences of vowel or non-vowel n-grams. The training procedure for the latter case is similar to that in the previous cases.

b) Phrase-based statistical machine translation

For this task, we concentrated on modifying the PSMT models in the following ways: firstly, we changed the training data in such a way that the set for tuning is part of the training set instead of keeping a separate set for tuning. In our basic setting, we remove 50 additional name pairs from the training set to be used for tuning the SMT model parameters. In another setting, we simply used them for training as well. Here, we were interested in seeing how increasing the training set influences the performance before training (especially with our tiny training set). Furthermore,

we would also like to know if tuning on parts of the training set may still lead to improvements on the test set.

Secondly, we changed the pre-processing step from character splitting to vowel/non-vowel splitting as described in the previous subsection for the WFST models. Here, we do not expect a similar effect on the results as we expect for the WFSTs. This is because contextual information is already integrated in the phrase-based SMT model to a large extent and important character combinations already appear in the extracted phrase table with appropriate scores.

A last modification we investigated is the application of a larger language model. It is well-known that SMT models produce better results in general when increasing the language model. However, the transliteration task is different from the sentence translation task for which the Europarl corpus is usually used. For the transliteration task, common character combinations in the target language may not necessarily be as common in named entities. Hence, we like to test the impact of adding data from a larger set of target language strings to estimate the character language model for our task.

7.5.3 Evaluation metrics

We use two metrics to evaluate the translations that are generated. The first measure which is commonly used is accuracy for which we compute the proportion of correctly transliterated names in the test set. Accuracy, as seen from the results in the previous task, is a very strict measure for character-based translation where one single mismatch is counted in the same way as a completely dissimilar pair of strings. Furthermore, for many transliterated names, several alternatives may be acceptable in a language (for example, "Chrushchev" instead of "Khrushchev") but only one reference is given in our data. In the previous task, we used an F-score measure that is based on the longest common subsequence between the candidate and reference transliterations. For this task, we use the longest common subsequence ratio (LCSR) as our main evaluation measure. Given a pair of strings, LCSR in this case is defined as the ratio of the length of the longest common subsequence and the length of the longer string. LCSR equal to 1 indicates a perfect match between the two strings.

7.5.4 Results

Let us first have a look at the baseline for this task. A common technique in machine translation for handling unknown words is to leave them untouched and to copy them to the target output. For names (usually a large portion of unknown words) this is certainly a good strategy if the writing system of the source and target language is

Method	Dut→Eng		Eng→Dut		Fre→Eng		Eng→Fre	
	LCSR	ACC	LCSR	ACC	LCSR	ACC	LCSR	ACC
Baseline	0.88	0.32	0.88	0.32	0.89	0.26	0.89	0.26
editWFST	0.88	0.22	0.87	0.20	**0.90**	**0.28**	0.89	0.24
1 state	0.88	0.22	0.87	0.18	**0.90**	0.26	0.89	0.24
2 states	0.79	0.00	0.88	0.18	0.79	0.02	0.88	0.14
3 states	0.81	0.12	0.80	0.04	0.85	0.14	0.81	0.00
4 states	0.81	0.06	0.85	0.22	0.78	0.02	0.81	0.02
5 states	0.78	0.02	0.78	0.02	0.79	0.04	0.83	0.04
vow/non-vow	0.83	0.20	0.84	0.28	0.88	**0.30**	0.87	0.20
phmm9wfst	0.88	0.22	0.87	0.18	**0.90**	0.26	0.89	0.24
phmm9wfstD+	0.88	0.32	0.88	0.26	**0.90**	**0.28**	0.89	0.24

Table 7.5: *LCSR and accuracy results associated with the use of weighted finite state transducers for character-based translation between English and Dutch, and between English and French.* ***Bolded*** *values indicate better performance over the baseline result.*

very similar. The baseline for our task refers to this strategy of copying the strings even for transliterated names.

Table 7.5 shows the translation results for the WFST models with those for the Baseline at the top. As we can see in Table 7.5, the LCSR baseline scores for both Dutch↔English and French↔English transliteration are quite high already, which means that Dutch and English, or French and English spellings of Russian names are not so different from each other. Even the accuracy is also high considering the strict nature of this measure. According to these results, the WFST models do not perform very well. None of the WFSTs actually improves the baseline LCSR nor accuracy for translation between Dutch and English. The translation performed by the WFSTs in this case would harm an SMT system that uses the baseline technique. For French→English, there is a slight improvement in LCSR and accuracy when the edit distance WFST is used. The use of Pair HMM parameters result in a slight improvement only on the LCSR measure for French→English translation. The vowel / non-vowel WFST model performs better than the baseline on the accuracy measure. We can also see that the edit distance WFST does not have a clear advantage over a single-state WFST for translation between the languages in the two language pairs. There is only a slight gain in accuracy for English→Dutch and French→English translation, otherwise the LCSR and ACC values are the same. It is also clear from Table 7.5 that the training procedure is not capable of learning a hidden underlying structure from the data. However, considering the size of our datasets, this should

not be expected. Looking at the large differences in the resulting LCSR and ACC for various numbers of states, it seems that the learning algorithm easily gets stuck in suboptimal maxima. Finally, the string splitting strategy of vowel/non-vowel sequences does not improve transliteration generation quality. On the contrary, it actually hurts the model, which is a bit surprising. One reason might be the increased sparseness of our dataset including larger sets of input and output symbols, which now contain character n-grams. The only improvement associated with the vowel/non-vowel WFST when compared with the other WFSTs can be seen for English→Dutch and French→English translation, albeit only on the accuracy measure. The accuracy for English→Dutch translation, is still below the baseline.

Table 7.6 shows the results from the phrase-based SMT system. We can see a clear improvement in the translation generation quality with regard to the LCSR measure. Except for the non-tuned models with large language models, all LCSR values are above the baseline LCSR value. The importance of training data can be seen in the values for translation between Dutch and English where the tuning set is included in the otherwise very small training data set. For these experiments, we obtain the highest LCSR and accuracy for translation in both directions. For translation between English and French where we have a larger training set, we do

PSMT	Dut→Eng		Eng→Dut		Fre→Eng		Eng→Fre	
	LCSR	ACC	LCSR	ACC	LCSR	ACC	LCSR	ACC
Baseline	0.88	0.32	0.88	0.32	0.89	0.26	0.89	0.26
without tuning	**0.89**	0.24	**0.90**	0.28	**0.91**	0.22	0.88	0.14
tuned	**0.92**	0.30	**0.90**	0.28	**0.93**	**0.46**	**0.91**	**0.28**
{tune} ⊂ {train}								
without tuning	**0.91**	**0.40**	**0.92**	0.32	**0.91**	**0.28**	0.89	0.18
tuned	**0.93**	**0.34**	**0.91**	**0.40**	**0.91**	**0.38**	**0.90**	**0.28**
vow/non-vow								
without tuning	**0.90**	0.28	**0.91**	**0.48**	**0.93**	**0.46**	**0.90**	0.28
tuned	**0.89**	0.32	**0.92**	**0.44**	**0.93**	**0.44**	**0.91**	**0.36**
large LM								
without tuning	0.82	0.06	0.82	0.06	0.76	0.02	0.84	0.02
tuned	**0.91**	0.26	**0.92**	**0.44**	**0.90**	0.22	**0.90**	0.22

Table 7.6: *LCSR and accuracy results associated with the use of a phrase-based statistical machine translation (PSMT) approach for a character-based translation between English and Dutch, and between Dutch and English. The descriptions for the PSMT models can be found in the text.* ***Bolded*** *values indicate better performance over the baseline result.*

not see a similar behavior. A separate development set seems to be preferable. Also, the impact of tuning is mixed and it is not clear how MERT is affected by a setting where the development set is not kept apart from training.

The strategy of splitting characters into vowel/non-vowel sequences makes the PSMT system perform quite well for English→Dutch translation. However, a clear advantage of this strategy over the standard pre-processing technique can not be seen.

In the final test, we included English, French and Dutch Europarl data (Koehn 2005) for estimating character-based language models (Table 7.6). We can clearly see that the additional data sets harm the translation process and only after tuning does LCSR and accuracy get back to the level of other models using the small language models from the parallel training data. Looking at the weights after tuning, we can also see that the language model weights are very low when using the large datasets. This seems to suggest that the overall influence of a language model on transliteration quality is rather low in our case.

Table 7.7 shows some examples of translations from the Dutch / English test set. In these examples, we can see typical problems especially of the WFST model. In particular, we can see the problem of consistent erroneous character substitutions without considering local context. For example, in the WFST translation, 'i' is consistently translated into 'i' in English and 'j' into 'y'. For the PSMT model, contextual dependencies are covered better due to the character n-grams in the translation table. However, there are still some ambiguities causing problems like '*tsjechov*'→'*chechov*' (instead of '*Chekhov*').

Dutch input	Correct English	WFST English	PSMT English
Andrej Tarkovski	Andrei Tarkovsky	Andrey Tarkovski	Andrey Tarkovsky
Anna Koernikova	Anna Kournikova	Anna Koernikova	Anna Kurnikova
Aleksandr Solzjenitsyn	Aleksandr Solzhenitsyn	Aleksandr Solzenitsyn	Alexandr Solzhenitsyn
Anton Tsjechov	Anton Chekhov	Anton Tsyekhov	Anton Chechov
Andrej Sacharov	Andrei Sakharov	Andrey Sakharov	Andrei Sakharov
Dmitri Sjostakovitsj	Dmitri Shostakovich	Dmitri Syostakovitsy	Dmitri Sjostakovich
Leonid Brezjnev	Leonid Brezhnev	Leonid Brezynev	Leonid Bruzhnev

Table 7.7: *Examples from the Dutch-English test set showing some typical problems of translating transliterations with the models.*

7.6 Conclusion

In this chapter, we have used the framework of weighted finite state automata to represent Pair HMMs for transliteration generation. The results associated with English→Russian transliteration generation suggested that the Pair HMM-based WFST models led to better transliteration generation quality compared to the usual application of WFSTs. However, the results also show that the performance from all the WFST models reported in this chapter is still lagging behind the state-of-the-art phrase-based statistical machine translation (PSMT) approach. This is obviously attributed to the lack of contextual information in the WFST models as compared to the case for the PSMT approach. On finding that the Pair HMM-based WFST models generated consistent mistransliterations after analyzing results from English→Russian transliteration, the use of some contextual information in the WFST models and the specification of a few contextual transformation rules in a post-processing step resulted in a large improvement in transliteration generation quality, but still below that of the PSMT-based system. The use of additional data also led to improved transliteration generation quality, with a larger improvement associated with the additional modifications to the Pair HMM-based WFST models.

We have also looked at the problem of translating transliterated names between languages that use the same writing system. We again applied WFST and PSMT-based models for transliteration generation between English and Dutch, and between English and French. We trained the models on name pairs of Russian origin extracted from Wikipedia. The PSMT-based approach performed best as expected, consistently beating the baseline of copying strings across the languages. The results in this case show that specialized models like the ones we have tested may help handle 'unknown' words in cross-lingual applications between languages using the same writing system when used in a transliteration generation framework.

We have only managed to use Pair HMM parameters in transliteration generation. The limitations that we associated with the Pair HMMs could be captured by the context-dependent transduction-based DBN models, however, we have not yet developed an interface that transforms the transduction-based DBN parameters to a format that is suitable for use in transliteration generation. The results suggested improved transliteration generation quality in regard to using contextual information in the Pair HMM - based WFSTs, it would be interesting to determine whether parameters from context-dependent DBN models (Chapter 5) could result in improved transliteration generation quality.

Chapter 8

Conclusions and future work

8.1 Conclusions

In this thesis, we have evaluated several edit distance-based Dynamic Bayesian Network (DBN) models for computing transliteration similarity in the tasks of transliteration detection and generation. We proposed to evaluate two edit distance-based DBN modeling approaches based on an observation of their successful applications in Natural Language Processing (NLP) tasks whose requirements are similar to those for transliteration detection. The first approach defines Pair Hidden Markov Models (Pair HMMs) which extend the classic Hidden Markov Models (HMMs) by enabling the representation of two observation sequences instead of one. The second approach which was initially developed for automatic speech recognition uses DBN templates to represent several types of transduction-based DBN models. The overall goal was to determine whether using models from the two edit distance-based DBN approaches for computing transliteration similarity could lead to improvements in transliteration detection and generation over state-of-the-art methods. In the following, we summarize the different stages of our investigation and present the resulting contributions.

We start with the first research question where we wanted to know whether existing methods for transliteration detection and generation suffice. To address this question, we have provided a literature review on transliteration detection and generation. In undertaking the literature review, we aimed for the following: defining the general setups used for the transliteration detection and generation tasks, identifying methods that have already been proposed and applied in the two tasks ranging from the earliest to current state-of-the-art, and establishing the necessity to apply the edit distance-based DBN models proposed in this thesis. As answers to the first research question, we found out the following. Firstly, that the large body of research on transliteration detection and generation included many attempts at addressing various factors that are important in modeling transliteration similarity. However,

based on recent evaluations of several systems, reports on the recent shared tasks on transliteration mining and generation (Li et al. 2009, Li et al. 2010, Kumaran et al. 2010b) showed that state-of-the-art methods are still far from achieving high accuracy. Secondly, that DBN models had already been used in the processes of detecting and generating transliterations but mostly in the form of the classic HMMs. To the best of our knowledge, there existed no report associated with using the edit distance-based DBN approaches proposed in this thesis in the two tasks of transliteration detection and generation.

For the second research question, we wanted to know whether there would be any gains in transliteration detection accuracy by following assumptions that were used to specify edit distance-based DBN models in the tasks where they had been successfully applied. If the assumptions from previous work did not hold for transliteration detection, we wanted to know whether modifications of the edit distance-based DBN models that meet requirements for computing transliteration similarity could lead to better transliteration detection quality. To address this research question, we first of all presented a conceptual framework describing our adaptations of different Pair HMMs (in Chapter 4) and transduction-based DBN models (in Chapter 5) for computing transliteration similarity. Based on ideas presented there, we evaluated the use of various parameterizations of the edit distance-based DBN models in experimental transliteration detection tasks using standard transliteration datasets. Results showed that Pair HMMs achieve considerable transliteration detection accuracy gains if we specify them to capture the differences in the writing systems of the source and target languages. More specifically, Chapter 4 results showed that Pair HMMs have a lower cross entropy (less uncertainty) on an English-Russian transliteration corpus if their emission parameters are based on the writing systems of the source and target language compared to being based on one writing system that combines the source and target language writing systems.

For the third research question, we wanted to know what features were critical in using the edit distance-based DBN models for computing transliteration similarity. To answer this research question, we began by investigating the effects of changing Pair HMM transition parameters on transliteration detection quality. Chapter 4 results showed that in addition to using emission parameters, it is important to use all standard Pair HMM transition parameters for computing transliteration similarity. In all transliteration detection experiments, we also evaluated the use of different Pair HMM algorithms for computing transliteration similarity. This includes: the Forward and Viterbi algorithms, and their log-odds versions obtained in combination with a random Pair HMM. Results showed that the log-odds versions of the Forward and Viterbi algorithms result in better transliteration detection quality compared to the base algorithms and that their transliteration detection performance is stable even

with changes in Pair HMM transition parameters. In Chapter 5, we adapted four generalizations of transduction-based DBN models initially proposed by Filali and Bilmes (2005) to compute transliteration similarity. Each generalization is used to define specific types of temporal dependencies which we hypothesize to be important for computing transliteration similarity. Three main classes of temporal dependencies include: edit operation memory dependencies; character context dependencies in source and / or target words; and edit operation length dependencies. We evaluated eight transduction-based DBN models using the same standard transliteration datasets that we used for the Pair HMMs. Results from transliteration detection experiments for seven language pairs showed that context-dependent DBN models perform better than the other transduction-based DBN models. A comparison with Pair HMMs showed that context-dependent DBN models perform better than Pair HMMs for 5/7 language pairs. The good performance of context-dependent DBN models here underlines the importance of representing context in DBN models for computing transliteration similarity. Despite the good transliteration detection performance from using context-dependent DBN models, results showed that there was still room for improvement. An error analysis on a sample of the transliteration detection results where atleast one context-dependent model failed to detect the correct transliterations at first rank, showed that improvements in transliteration detection accuracy would result if we combined the models. Based on this observation, a simple combination scheme of using the average transliteration similarity estimates of the combined DBN models resulted in improvements in transliteration detection accuracy over 'individual' application of the DBN models.

For the transliteration generation task, we have also provided a conceptual framework for representing Pair HMMs as Weighted Finite State Transducers (WFSTs) which enable an evaluation of using Pair HMM parameters for suggesting candidate transliterations. The Pair HMM-based WFSTs, however, are disadvantaged by the lack of source and / or target character contextual information in their parameters compared to state-of-the-art transliteration generation methods such as phrase-based statistical machine transliteration (PSMT) (Matthews 2007, Finch and Sumita 2008) and approaches that first induce many-to-many alignments (Jiampojamarn et al. 2010). It is not surprising that the Pair HMM-based WFSTs did not perform better than the state-of-the-art methods which used a lot of character context information in their models. An error analysis on a sample of English→Russian transliteration generation results from development data showed consistent mistransliterations from using Pair HMM parameters. A post-processing step where we used a few transformation rules aimed at dealing with the consistent mistransliterations resulted in improved transliteration generation quality on test data. Also, a pre-specification of identified character combinations aimed at representing some contextual informa-

tion in the Pair HMM parameters led to improved English→Russian transliteration generation quality to the extent that it approached that for state-of-the-art methods mentioned above.

For the fourth research question, we wanted to know whether using the edit distance-based DBN models could improve transliteration mining and generation quality compared to state-of-the-art methods. To address this question, we first evaluated a selection of Pair HMMs and context-dependent DBN models against state-of-the-art approaches in mining transliterations from standard Wikipedia paired topics which were provided as standard corpora for evaluating language-independent systems in the NEWS 2010 shared tasks on transliteration mining (Kumaran et al. 2010b). Chapter 6 results suggest that the transliteration mining performance from using the edit distance-based DBN models is comparable to that for state-of-the-art methods (Nabende 2010c). Chapter 6 results also show considerable improvements in transliteration mining quality from using Pair HMMs over state-of-the-art approaches on transliteration corpora for two language pairs: English-Hindi and English-Tamil. In addition to using only Wikipedia paired topics, we also proposed to apply the DBN models in mining transliterations from the main content of comparable, bilingual Wikipedia pages. Based on the premise that the number of words contained in the article content exceed the number of words in the topics by a multiple, we expected increased coverage by extracting transliteration pairs from the article content. Research on mining transliterations from Wikipedia commonly involves the use of training data that is prepared from an external source. In our case, we also show that it is possible to use only Wikipedia data for mining transliterations. Specifically, we have applied a method from related work where we search for only particular types of Wikipedia topics such as *person names* which are in most cases standard translations across languages and used them as training data. Chapter 6 results suggested a promising application of the Pair HMMs and context-dependent DBN models in mining transliterations from the very 'noisy' comparable Wikipedia article content.

Finally, previous work on transliteration mining and generation emphasizes mapping from one writing system to another. However, cross-language processing between languages that use the same writing system, with the presence of *unknown* entities, is affected in a manner similar to the case where the languages use different writing systems. We therefore proposed the usual application of the transliteration mining and generation framework to that where the languages use the same writing system. To the best of our knowledge, this task had not yet been addressed in transliteration mining and generation literature. After testing various models, Chapter 7 results show that the usual transliteration generation setup leads to considerable accuracy gains over the standard baseline of copying strings from the source language to the target language.

8.2 Future work

Although our conclusions show a valuable application of DBN models in transliteration mining, a number of interesting research directions can follow from the work presented in the thesis. First of all, we begin with what is unfinished.

For the transliteration mining task, we proposed an investigation into several DBN model settings. But as is described in the thesis, the DBN approaches offer a limitless model space for which we have not exhaustively explored. For the edit distance based DBN modeling approach in particular, we propose an investigation into the application of additional models. We also evaluated the DBN models against the state-of-the-art methods on only five language pairs. It should be interesting to evaluate their performance on real-world data for additional language pairs.

For the transliteration generation task, we have only investigated the use of Pair HMM parameters in transliteration generation. One interesting research direction is to investigate the use of the parameters for the transduction-based DBN models in the transliteration generation task. Since the transduction-based DBN models are based on a representation of a stochastic memoryless transducer, we postulate that a transformation of the DBN models to finite state automata representations should be possible so as to enable the evaluation of the use of DBN model parameters for transliteration generation.

There are other research directions that can be followed in addition to our unfinished work above. Here, we would like to note that our investigation into the use of the transduction-based DBN models was mostly affected by processing speed during inference based on the models. We have reported in the use of the *Frontier algorithm*, but there are other algorithms that have been proposed to improve computational efficiency while maintaining and in some cases improving effectiveness in using DBN models. It should be interesting to investigate the use of other inference algorithms for the transduction-based DBN models.

Our work is just but an additional application of the two DBN approaches in machine transliteration. Our successful application of the DBN models in mining transliterations from noisy Wikipedia data suggests the application of the DBN models to problems where there is need to handle 'noise' in sequences. It should also be interesting to investigate the application of the DBN approaches presented in this thesis to address a variety of problems based on edit operations.

Bibliography

Abduljaleel, N. and Larkey, L.: 2003, Statistical transliteration for English-Arabic cross language information retrieval, *Proceedings of the twelfth international conference on Information and knowledge management*, ACM, New Orleans, LA, USA, pp. 139–146.

Adafre, S. and de Rijke, M.: 2006, Finding similar sentences across multiple languages, *Proceedings of the EACL Workshop on NEW TEXT Wikis and blogs and other dynamic text sources*, Trento, Italy, pp. 62–69.

Al-Onaizan, Y. and Knight, K.: 2002, Machine transliteration of names in Arabic text, *Proceedings of the ACL-02 workshop on Computational approaches to semitic languages*, pp. 1–13.

Arbabi, M., Fischtal, S., Cheng, V. and Bart, E.: 1994, Algorithms for Arabic name transliteration, *IBM J. Res. Develop.* **38**(2), 183–193.

Arfken, G.: 1985, *Mathematical methods for physicists*, Academic Press.

Arribas-Gil, A., Gassiat, E. and Matias, C.: 2006, Parameter estimation in Pair-hidden Markov Models, *Scandinavian Journal of Statistics* **33**(4), 651–671.

Baum, L.: 1972, An inequality and associated maximization technique in statistical estimation of probabilistic functions of a Markov process, *Inequalities* **3**.

Baum, L., Petrie, T., Soules, G. and Weiss, N.: 1970, A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains, *The Annals of Mathematical Statistics* **41**(1), 164–171.

Bilmes, J.: 2002, The graphical models toolkit – Documentation, Technical Documentation on the Web.

Bilmes, J. and Zweig, G.: 2002, The graphical models toolkit: an open source software system for speech and time-series processing, *Proceedings of IEEE International conference on Acoustics, Speech, and Signal processing*.

Borman, S.: 2004, The Expectation Maximization algorithm – A short tutorial. Inotroduces the Expectation Maximization (EM) algorithm and fleshes out the basic mathematical results, including a proof of convergence. The Generalized EM algorithm is also introduced.

Bouma, G., Duarte, S. and Islam, Z.: 2009, Cross-lingual alignment and completion of Wikipedia Templates, *Proceedings of the third International workshop on cross lingual information access: addressing the need of multilingual societies*, Boulder, Colorado, pp. 21–29.

Boyen, X., Friedman, N. and Koller, D.: 1999, Discovering the hidden structure of complex dynamic systems, *Proceedings of the 15th annual conference on Uncertainty in AI (UAI)*, pp. 206–215.

Brown, P., Pietra, S., Pietra, V. and Mercer, R.: 1993, The mathematics of statistical machine translation: parameter estimation, *Computational Linguistics* **19**(2), 263–311.

Chen, K.-J. and Bai, M.-H.: 1998, Unknown word detection for chinese by a Corpus-based Learning Method, *Computational Linguistics and Chinese Language Processing* **3**(1), 27–44.

Chinnakotla, M., Damani, O. and Satoskar, A.: 2010, Transliteration for resource-scarce languages, *ACM Transactions on Asian Language Information Processing* **9**(4), article 14.

Covington, M.: 1996, An algorithm to align words for historical comparison, *Computational Linguistics* **22**(4), 481–496.

Darwiche, A.: 2008, Bayesian networks, *in* F. v. Harmelen, V. Lifschitz and B. Porter (eds), *Handbook of Knowledge Representation*, Vol. 3 of *Foundations of Artificial Intelligence*, Elsevier, pp. 467–509.

Darwish, K.: 2010, Transliteration mining with phonetic conflation and iterative training, *Proceedings of the 2010 Named Entities Workshop, ACL 2010*, Association for Computational Linguistics, Uppsala, Sweden, pp. 53–56.

Das, A., Saikh, T., Mondal, T., Ekbal, A. and Bandyopadhyay, S.: 2010, English to Indian languages machine transliteration system at NEWS 2010, *Proceedings of the 2010 Named Entities Workshop*, Uppsala, Sweden, pp. 71–75.

Dijkstra, E.: 1959, A note on two problems in connexion with graphs, *Numerische Mathematik* **1**.

Durbin, R., Eddy, S., Krogh, A. and Mitchison, G.: 1998, *Biological sequence analysis: Probabilistic models of proteins and nucleic acids*, Cambridge University Press, Cambridge, UK.

Ekbal, A., Naskar, S. and Bandyopadhyay, S.: 2006, A modified joint source-channel model for transliteration, *Proceedings of COLING/ACL*, Sydney, Australia, pp. 191–198.

Erdmann, M., Nakayama, K., Hara, T. and Nishio, S.: 2008, An approach for extracting bilingual terminology from wikipedia, *Proceedings of the 13th international conference on Database Systems for advanced applications*, New Delhi, India, pp. 380–392.

Federico, M., Bertoldi, N. and Cetolo, M.: 2008, Irstlm: An open source toolkit for handling large scale language models, *Proceedings of Interspeech*, Melbourne, Australia, pp. 1618–1621.

Filali, K. and Bilmes, J.: 2005, A Dynamic Bayesian framework to model context and memory in edit distance learning: an application to pronunciation classification, *Proceedings of the Association for Computational Linguistics (ACL)*, Ann-Arbor, Michigan.

Finch, A. and Sumita, E.: 2008, Phrase-based machine transliteration, *Proceedings of the Workshop on Technologies and corpora for Asia-Pacific speech translation*, Hyderabad, India, pp. 13–18.

Finch, A. and Sumita, E.: 2010, Transliteration using a phrase-based statistical machine translation system to re-score the output of a joint multigram model, *Proceedings of the 2010 Named Entities Workshop*, Uppsala, Sweden, pp. 48–52.

Freeman, A., Condon, S. and Ackerman, C.: 2006, Cross linguistic name matching in english and arabic, *Proceedings of the Human Language Technology conference of the NAACL*, New York city, USA, pp. 471–478.

Gao, W., Wong, K.-F. and Lam, W.: 2005, Phoneme-based transliteration of foreign names for OOV problem, *IJCNLP, LNAI 3248*, Springer-Verlag, Berlin Heidelberg, pp. 110–119.

Ghahramani, Z.: 1998, Learning dynamic bayesian networks, *Adaptive Processing of Sequences and Data Structures*, Springer-Verlag, pp. 168–197.

Ghahramani, Z.: 2001, An introduction to Hidden Markov models and Bayesian networks, *IJPRAI* **15**(1), 9–42.

Goto, I., Kato, N., Uratani, N. and Ehara, T.: 2003, Transliteration considering context information based on the maximum entropy method, *Proceedings of MT-Summit IX*, pp. 125–132.

He, X.: 2007, Using word dependent transition models in HMM based word alignment for statistical machine translation, *Proceedings of 2nd ACL workshop on statistical machine translation*, Prague, Czech Republic, pp. 80–87.

Hetland, M. L.: 2004, *Data mining in time series databases*, World scientific.

Jensen, F. and Nielsen, T.: 2007, *Bayesian networks and decision graphs*, Springer Publishing company, Incorporated.

Jensen, J.: 1906, Sur les fonctions convexes et les inégalités entre les valeurs moyennes, *Acta Mathematica* **130**(1), 175–193.

Jeong, K., Myaeng, S., Lee, J. and Choi, K.-S.: 1999, Automatic identification and back-transliteration of foreign words for information retrieval, *Information Processing and Management* **35**(4), 523–540.

Jiampojamarn, S., Bhargava, A., Dou, Q., Dwyer, K. and Kondrak, G.: 2009, DirectL: a language-independent approach to transliteration, *Proceedings of the 2009 Named Entities Workshop: Shared task on transliteration*, pp. 28–31.

Jiampojamarn, S., Dwyer, K., Bergsma, S., Bhargava, A., Dou, Q., Kim, M.-Y. and Kondrak, G.: 2010, Transliteration generation and mining with limited training resources, *Proceedings of the 2010 Named Entities Workshop*, Association for Computational Linguistics, Uppsala, Sweden, pp. 39–47.

Jung, S., Hong, S. and Paek, E.: 2000, An English to Korean transliteration model of extended markov window, *Proceedings of the 18th conference on Computational Linguistics*, Saarbrucken, Germany, pp. 383–389.

Jurafsky, D. and Martin, J.: 2009, *Speech and language processing*, Pearson Education, Inc., Upper Saddle River, New Jersey.

Kang, B. and Choi, K.-S.: 2000, Automatic transliteration and back-transliteration by decision tree learning, *Proceedings of the 2nd International Conference on Language Resources and Evaluation*, pp. 1135–1411.

Kang, I. and Kim, G.: 2000, English-to-Korean transliteration using multiple unbounded overlapping phoneme chunks, *Proceedings of the 18th International conference on Computational Linguistics*, pp. 418–424.

Karimi, S., Scholar, F. and Turpin, A.: 2011, Machine transliteration survey, *ACM Comput. Surv.* **43**(3), 17:1–17:46.

Karimi, S., Turpin, A. and Scholer, F.: 2006, English to Persian transliteration, *in* F. Crestani, P. Ferragina and M. Sanderson (eds), *String Processing and Information Retrieval*, Vol. 4209 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 255–266.

Karimi, S., Turpin, A. and Scholer, F.: 2007, Collapsed consonant vowel models: new approaches for English-Persian transliteration and back-transliteration, *Proceedings of the 45th Annual meeting of the Association of Computational Linguistics*, Czech Republic, pp. 648–655.

Kawtrakul, A., Deemagarn, A., Thumkanon, C., Khantonthong, N. and McFetridge, P.: 1998, Backward transliteration for Thai document retrieval, *Proceedings of IEEE Asia Pacific Conference on Circuits and Systems*, Chiang-mai, Thailand, pp. 563–566.

Khapra, M., Udupa, R., Kumaran, A. and Bhattacharyya, P.: 2010, PR + RQ almost equal to PQ: Transliteration mining using Bridge language, *AAAI*, Atlanta, Georgia, USA, pp. 1346–1351.

Kim, J., Lee, J. and Choi, K.: 1999, Pronunciation unit based automatic English-Korean transliteration model using neural network, *Proceedings of Korea Cognitive Science Association*, pp. 247–252.

Kirschenbaum, A. and Wintner, S.: 2010, A general method for creating a bilingual transliteration dictionary, *Proceedings of the seventh international conference on Language Resources and Evaluation Conference (LREC-2010)*, Valletta, Malta, pp. 3389–3392.

Klementiev, A. and Roth, D.: 2006, Named entity transliteration and discovery from multilingual comparable corpora, *Proceedings of the Human Languages Technology Conference of the NAACL, Main Conference*, New York City, USA, pp. 82–88.

Knight, K. and Graehl, J.: 1997, Machine transliteration, *Proceedings of the 35th Annual meeting of the ACL and Eighth Conference of the European chapter of the ACL*, Madrid, Spain, pp. 128–135.

Koehn, P.: 2005, Europarl: A parallel corpus for statistical machine translation, *Conference proceedings: the tenth Machine Translation summit*, Phuket, Thailand, pp. 79–86.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A. and Herbst, E.: 2007, Moses: open source toolkit for statistical machine translation, *Proceedings of the 45th Annual meeting of the ACL*, Prague, Czech Republic, pp. 177–180.

Koehn, P., Och, F., Franz, J. and Marcu, D.: 2003, Statistical phrase-based translation, *Proceedings of the 2003 conference on the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, Edmonton, Canada, pp. 48–54.

Koller, D. and Friedman, N.: 2009, *Probabilistic Graphical models: Principles and Techniques*, MIT Press.

Kondrak, G. and Sherif, T.: 2006, Evaluation of several phonetic similarity algorithms on the task of cognate identification, *Proceedings of the Workshop on Linguistic Distances*, LD '06, Sydney, Australia, pp. 43–50.

Kumaran, A., Khapra, M. and Li, H.: 2010a, Report of NEWS 2010 shared task on transliteration mining, *Proceedings of the 2010 Named Entities Workshop*, Association for Computational Linguistics, Uppsala, Sweden, pp. 21–28.

Kumaran, A., Khapra, M. and Li, H.: 2010b, Whitepaper of NEWS 2010 shared task on transliteration mining, *Proceedings of the 2010 Named Entities Workshop*, Association for Computational Linguistics, Uppsala, Sweden, pp. 29–38.

Kuo, J.-S., Li, H. and Yang, Y.-K.: 2007, A phonetic similarity model for automatic extraction of transliteration pairs, *ACM Trans. Asian Language Information Processing* **6**(2), article 6.

Lafferty, J., McCallum, A. and Pereira, F.: 2001, Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data, *Proceedings of the 18th international conference on machine learning*, Williamstown, MA, USA, pp. 282–289.

Lee, C.-J. and Chang, J.: 2003, Acquisition of English-Chinese transliterated word pairs from parallel-aligned texts using a statistical machine transliteration model, *Proceedings of the HLT-NAACL 2003 Workshop on Building and using parallel texts: data driven machine translation and beyond*, HLT-NAACL-PARALLEL 'O3, Edmonton, Canada, pp. 96–103.

Lee, J. and Choi, K.-S.: 1998, English to Korean statistical transliteration for Information Retrieval, *Computer Processing of Oriental Languages* **12**(1), 17–37.

Li, H., Kumaran, A., Pervouchine, V. and Zhang, M.: 2009, Report of NEWS 2009 machine transliteration shared task, *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration*, Suntec, Singapore, pp. 1–18.

Li, H., Kumaran, A., Zhang, M. and Pervouchine, V.: 2010, Report of NEWS 2010 transliteration generation shared task, *Proceedings of the 2010 Named Entities Workshop*, Association for Computational Linguistics, Uppsala, Sweden.

Li, H., Sim, K., Kuo, J.-S. and Dong, M.: 2007, Semantic transliteration of personal names, *Proceedings of the 45th Annual meeting of the association of computational linguistics*, Prague, Czech Republic, pp. 120–127.

Li, H., Zhang, M. and Su, J.: 2004, A joint source-channel model for machine transliteration, *Proceedings of the 42nd meeting of the Association for Computational Linguistics (ACL '04)*, Barcelona, Spain, pp. 159–166.

Lin, T., Wu, J.-C. and Chang, J.: 2004, Extraction of name and transliteration in monolingual and parallel corpora, *AMTA*, pp. 177–186.

Lindén, K.: 2006, Multilingual modeling of cross-lingual spelling variants, *Information Retrieval* **9**(3), 295–310.

Mackay, W.: 2004, *Word similarity using Pair HMMs*, Master's thesis, University of Alberta, Alberta, Canada.

Mackay, W. and Kondrak, G.: 2005, Computing word similarity and identifying cognates with Pair Hidden Markov models, *Proceedings of the ninth Conference on Computational Natural Language Learning (CONLL 2005)*, Ann Arbor, Michigan, pp. 40–47.

Malik, M.: 2006, Punjabi machine transliteration, *Proceedings of the 21st International conference on computational linguistics*, Sydney, Australia, pp. 1137–1144.

Malik, M., Boitet, C. and Bhattacharyya, P.: 2008, Hindi Urdu machine transliteration using finite-state transducers, *Proceedings of the 22nd International Conference on Computational Linguistics*, Manchester, United Kingdom, pp. 537–544.

Manning, C. and Schütze, H.: 1999, *Foundations of statistical Natural Language Processing*, MIT Press, Cambridge, MA.

Matthews, D.: 2007, *Machine transliteration of proper names*, Master's thesis, School of Informatics, University of Edinburgh.

Meng, H., Wai-Kit, L., Berlin, C. and Tang, K.: 2001, Generating phonetic cognates to handle named entities in English-Chinese cross-language spoken document retrieval, *Proceedings of IEEE workshop on Automatic Speech Recognition and Understanding*, Madonna di Campiglio, Italy, pp. 311–314.

Mihajlovic, V. and Petkovic, M.: 2001, Dynamic Bayesian Networks: A state of the art. DMW-Project.

Mohammadi, M. and GhasemAghaee, N.: 2010, Building bilingual parallel corpora based on Wikipedia, *International Conference on Computer Engineering and Applications* **2**, 264–268.

Murphy, K.: 2002, *Dynamic Bayesian Networks: Representation, Inference and Learning*, PhD thesis, UC Berkeley, Computer Science Division.

Nabende, P.: 2009a, Evaluation of Dynamic Bayesian Network models for entity name transliteration, *Proceedings of the 18th annual Belgian-Dutch Conference on machine learning*, Tilburg, The Netherlands, pp. 99–100.

Nabende, P.: 2009b, Transliteration system using Pair HMM with weighted FSTs, *Proceedings of the 2009 named entities workshop: shared task on transliteration*, Suntec, Singapore, pp. 100–103.

Nabende, P.: 2010a, Applying a Dynamic Bayesian Network framework to transliteration identification, *Proceedings of the 7th International Conference on Language Resources and Evaluation*, Valletta, Malta, pp. 244–251.

Nabende, P.: 2010b, Comparison of applying Pair HHMs and DBN models in transliteration identification, *Proceedings of the 20th Computational Linguistics in Netherlands conference*, Utrecht, The Netherlands, pp. 107–122.

Nabende, P.: 2010c, Mining transliterations from Wikipedia using pair HMMs, *Proceedings of the 2010 Named Entities Workshop*, Uppsala, Sweden, pp. 76–80.

Nabende, P.: 2011, Mining transliterations from Wikipedia using Dynamic Bayesian Networks, *Proceedings of the international conference on recent advances in natural language processing*, Hissar, Bulgaria, pp. 385–391.

Nabende, P., Tiedemann, J. and Nerbonne, J.: 2010, Pair Hidden Markov Model for named entity matching, *Innovations and Advances in Computer Sciences and Engineering*, Springer, Heildelberg, pp. 497–502.

Noeman, S. and Madkour, A.: 2010, Language independent transliteration mining system using finite state automata framework, *Proceedings of the 2010 Named Entities Workshop*, Uppsala, Sweden.

Och, J. and Ney, H.: 2003, A systematic comparison of various statistical alignment models, *Computational Linguistics* **29**(1), 19–51.

Oh, J.-H. and Choi, K.-S.: 2002, An English-Korean transliteration model using pronunciation and contextual rules, *Proceedings of the 19th International conference on computational linguistics*, COLING '02, Taipei, Taiwan, pp. 1–7.

Oh, J.-H., Choi, K.-S. and Isahara, H.: 2006, A comparison of different machine transliteration models, *Journal of artificial intelligence research* **27**(2006), 119–151.

Oh, J., Uchimoto, K. and Torisawa, K.: 2009, Machine transliteration using target-language grapheme and phoneme: multi-engine transliteration approach, *Proceedings of the 2009 Named Entites Workshop: Shared Task on Transliteration*, pp. 36–39.

Quinlan, J.: 1986, Induction Decision trees, *Machine Learning* **1**(1), 81–106.

Rabiner, L.: 1989, A tutorial on Hidden Markov models and selected applications in Speech Recognition, *Proceedings of the IEEE*, pp. 257–286.

Ristad, E. and Yianilos, P.: 1997, Learning string edit distance, *IEEE Transactions on Pattern analysis and machine intelligence* **20**, 522–532.

Roger, M.: 1992, *A description of a computer-usable dictionary file based on the Oxford advanced learner's dictionary of current English*, Oxford Text Archive.

Rosenblatt, F.: 1958, A probabilistic model for information storage and organization in the brain, *Psychological review* **65**.

Shannon, C.: 1948, A mathematical theory of communication, *Bell System Technical Journal* **27**, 379–423, 623–656.

Sherif, T. and Kondrak, G.: 2007a, Bootstrapping a stochastic transducer for Arabic-English transliteration extraction, *Proceedings of the 45th Annual meeting of the Association of Computational Linguistics*, Prague, Czech Republic, pp. 864–871.

Sherif, T. and Kondrak, G.: 2007b, Substring-based transliteration, *Proceedings of the 45th annual meeting of the Association for Computational Linguistics*, Prague, Czech Republic, pp. 944–951.

Song, Y., Kit, C. and Zhao, H.: 2010, Reranking with multiple features for better transliteration, *Proceedings of the 2010 Named Entities Workshop*, Uppsala, Sweden, pp. 62–65.

Sproat, R., Tao, T. and Zhai, C.: 2006, Named entity transliteration with comparable corpora, *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, Sydney, Australia, pp. 73–80.

Stalls, B. and Knight, K.: 1998, Translating names and technical terms in Arabic text, *Proceedings of the COLING/ACL Workshop on computational approaches to semitic languages*, pp. 34–41.

Surana, H. and Singh, A.: 2008, A more discerning and adaptable multilingual transliteration mechanism for Indian languages, *Proceedings of the third International Joint Conference on Natural Language Processing (IJCNLP)*, Asian Federation of Natural Language Processing, Hyderabad, India.

Tiedemann, J. and Nabende, P.: 2009, Translating transliterations, *International journal of computing and ICT research* **3**(1), 33–41.

Udupa, R., Saravanan, K., Kumaran, A. and Jagarlamudi, J.: 2009, MINT: A method for effective and scalable mining of named entity transliterations from large comparable corpora, *Proceedings of the 12th Conference of the European Chapter of the ACL*, Athens, Greece, pp. 799–807.

Virga, P. and Khudanpur, S.: 2003, Transliteration of proper names in cross-lingual information retrieval, *Proceedings of the ACL 2003 Workshop on multiple and mixed language named entity recognition*, Sapporo, Japan, pp. 57–64.

Vogel, S., Ney, H. and Tillmann, C.: 1996, HMM-based word alignment in statistical translation, *Proceedings of the 16th conference on computational linguistics*, Copenhagen, Denmark, pp. 836–841.

Wan, S. and Verspoor, C.: 1998, Automatic English-Chinese name transliteration for development of multilingual resources, *Proceedings of the 36th annual meeting of the Association for computational linguistics*, Montreal, Quebec, Canada, pp. 1352–1356.

Wieling, M., Leinonen, T. and Nerbonne, J.: 2007, Inducing sound segment differences using Pair Hidden Markov models, *in* J. Nerbonne, M. Ellison and G. Kondrak (eds), *Computing and Historical Phonology: 9th Meeting of ACL Special Interest Group for Computational Morphology and Phonology Workshop*, Prague, pp. 48–56.

Zelenko, D. and Aone, C.: 2006, Discriminative methods for transliteration, *Proceedings of the 2006 conference on Empirical methods in Natural Language Processing*, Sydney, Australia, pp. 612–617.

Zhang, M., Kumaran, A. and Li, H.: 2011, Whitepaper of NEWS 2011 shared task on machine transliteration, *Proceedings of the 2011 Named Entities Workshop*, Chiang Mai, Thailand.

Zhang, M., Li, H. and Su, J.: 2004, Direct orthographical mapping for machine transliteration, *Proceedings of the 20th international conference on computational linguistics*, COLING 'O4, Geneva, Switzerland, p. Article 716.

Zhao, B., Bach, N., Lane, I. and Vogel, S.: 2007, A log-linear block transliteration model based on bi-stream HMMs, *Proceedings of NAACL HLT 2007*, Rochester, NY, pp. 364–371.

Zhou, Y.: 2009, Maximum n-gram HMM-based name transliteration: Experiment in NEWS 2009 on English-Chinese corpus, *Proceedings of the 2009 Named Entities workshop, ACL-IJCNLP*, Suntec, Singapore.

Zweig, G.: 1996, *A forward-backward algorithm for inference in Bayesian networks and an empirical comparison with HMMs*, Master's thesis, Dept. Comp. Sci., U.C. Berkeley.

Zweig, G. and Russell, S.: 1999, Probabilistic modeling with Bayesian networks for Automatic Speech Recognition, *Australian Journal of Intelligent Information Processing Systems* **5**(4), 253–260.

Index

Summary

Automated transliteration detection and generation are Natural Language Processing (NLP) tasks aimed at improving performance in related NLP applications such as Machine Translation (MT) and Cross Language Information Retrieval (CLIR). Currently, there is a growing body of research on identifying methods that can improve transliteration detection and generation quality including annual shared tasks (Li et al. 2009, Li et al. 2010, Kumaran et al. 2010b, Zhang et al. 2011). In this thesis, we propose applying two edit distance-based approaches of Dynamic Bayesian Network (DBN) models for computing transliteration similarity in the two tasks of transliteration detection and generation. The first approach uses Pair Hidden Markov Models (Pair HMMs) which are a generalization of the classic Hidden Markov models (HMMs). The second approach, which is inspired by work on Automatic Speech Recognition (ASR), uses DBN templates to define several classes of transduction-based DBN models. The Pair HMMs and the transduction-based DBN models have been applied successfully in NLP tasks that have requirements similar to those for transliteration detection and generation. The Pair HMMs were applied successfully in cognate identification (Mackay and Kondrak 2005) and dialect comparison (Wieling et al. 2007), whereas the tranduction-based DBN models were applied successfully in pronunciation classification (Filali and Bilmes 2005) and cognate identification (Kondrak and Sherif 2006).

We explore the effects of several factors represented by the DBN models on

transliteration detection and generation quality. From the approach of Pair HMMs, we first evaluate assumptions that were previously proposed and used in the NLP tasks mentioned above. That is, we evaluate the use of Pair HMM parametric structures similar to those used in previous work. For the transliteration detection experiments at this stage, we use our own pre-processed transliteration data consisting of geographic name pairs extracted from the online Geonames database for four language pairs: English-Dutch, English-French, English-German, and English-Russian. As the language pair names suggest, we explore the use of the traditional transliteration detection and generation setups in cases where source and target language use the same writing system; to the best of our knowledge, this had not yet been addressed in machine transliteration literature. Results at this preliminary stage suggest the importance of using Pair HMMs that adequately capture the differences in the writing systems of source and target languages. For example, that the Pair HMM emission parameters should be based on the writing systems of the source and target language rather than being based on one writing sytem that combines the source and target language writing systems. We then investigate the effect of changes in Pair HMM transition parameters on transliteration detection and generation quality. Each Pair HMM transition parameter encodes the probability of transition from one edit operation state to another or back to the same state. In the transliteration detection experiments, we evaluate the performance of the Pair HMMs against a standard baseline of using pair n-gram information for computing transliteration similarity. Here, we use standard transliteration data from the 2009 and 2010 Named Entities Workshop (NEWS) shared tasks on transliteration generation for seven language pairs: English-Bengali, English-Chinese, English-Hindi, English-Kannada, English-Russian, English-Tamil, and English-Thai. Results suggest that the use of Pair HMM transition parameters results in considerably better transliteration detection quality than when no transition parameters are used. In all transliteration detection experiments for the Pair HMMs, we evaluate the use of different algorithms for computing transliteration similarity including: the Forward and Viterbi algorithms, and their log-odds versions obtained in combination with a random Pair HMM. Results show that the log-odds versions of the Forward and Viterbi algorithms result in better transliteration detection quality compared to the base algorithms and that their transliteration detection performance is stable even with changes in transition parameters. The Pair HMMs also post the best transliteration detection accuracy and Mean Reciprocal Rank (MRR) results compared to the standard baseline of Pair n-gram information for six of the seven language pairs. The results at this point suggest that Pair HMMs could be valuable in searching for transliterations.

The second approach of DBN models is founded on the representation of a mem-

oryless stochastic transducer from Ristad and Yianilos (1997) as a Memoryless and Context-Independent (MCI) DBN model. We adapt the MCI DBN model from Filali and Bilmes (2005) to compute transliteration similarity for transliteration detection. Based on the MCI DBN model we adapt and evaluate the use of three generalizations of transduction-based DBN models. Each generalization is used to define specific types of temporal dependencies which we hypothesize to be important for computing transliteration similarity. The three classes of temporal dependencies include: edit operation memory dependencies; character context dependencies in the source and / or target words; and edit operation length dependencies. We use standard transliteration data from the 2009 and 2010 shared tasks on transliteration generation as described in the previous paragraph to evaluate eight transduction-based DBN models for computing transliteration similarity in the same experimental setup of transliteration detection as used for the Pair HMMs. Results from the transliteration detection experiments for the seven language pairs show that context-dependent DBN models perform better than the other transduction-based DBN models. A comparison with Pair HMMs shows that context-dependent DBN models outperform Pair HMMs for 5/7 language pairs. The context-dependent DBN models also outperform the standard baseline of using Pair n-gram information for all language pairs. The results here underline the importance of representing context in DBN models for computing transliteration similarity. An error analysis associated with the application of transduction-based DBN models showed that there was still room for improvement in transliteration detection quality if there is a way to combine the application of the DBN models. Based on that, we applied a simple combination scheme of using the average transliteration similarity estimate computed from the transliteration similarity estimates of the combined DBN models for detecting transliteration pairs. We evaluated several combinations of transliteration similarity estimates from context-dependent DBN models and results showed improved transliteration detection accuracy and MRR over 'individual' application of the DBN models.

Based on the aim of further establishing the value of using DBN models to compute transliteration similarity for transliteration detection and generation, we evaluated the performance of some Pair HMMs and context-dependent DBN models against state-of-the-art methods by following the same evaluation setup used in the 2009 and 2010 shared tasks on transliteration generation (Li et al. 2009, Li et al. 2010) and transliteration mining (Kumaran et al. 2010a). For the transliteration mining task, we applied Pair HMMs and context-dependent DBN models in mining single word transliteration pairs from standard Wikipedia topic pairs obtained from Wikipedia inter-language links for five language pairs: English-Arabic, English-Chinese, English-Hindi, English-Russian, and English-Tamil. Results show that the Pair HMMs using log-odds ratio to compute transliteration similarity con-

siderably improve f-score values for mining English-Hindi and English-Tamil transliterations compared to the best results from state-of-the-art systems. The Pair HMMs and context-dependent DBN models also posted competitive f-score values for mining English-Arabic, English-Chinese, and English-Russian transliteration pairs. For transliteration generation, we evaluated the use of Pair HMM parameters as weighted transduction parameters for generating candidate transliterations given a 'source language' word. Here, we proposed Weighted Finite State Transducer (WFST) models to approximate corresponding Pair HMMs. We evaluated the Pair HMM-based WFST models against state-of-the-art methods using standard transliteration data from the 2009 and 2010 shared tasks on transliteration generation (Li et al. 2009, Li et al. 2010) as described above. Results show that the transliteration generation quality of the Pair HMM-based WFSTs on all language pairs still lags behind that of state-of-the-art approaches such as Phrase-based statistical machine transliteration (PSMT). One disadvantage of the Pair HMM-based WFST models as compared to state-of-the-art approaches is the lack of representation for source and / or target character contextual information. From an error analysis on the results of the transliteration generation task for English to Russian, we found out that the Pair HMM-based WFSTs resulted in consistent mistransliterations. A specification of a few contextual transformation rules in a post-processing step resulted in a large improvement in transliteration generation quality but still below that of state-of-the-art approaches like the PSMT approach.

Samenvatting

Automatische transliteratie en generatie zijn nuttig voor het verbeteren van de prestatie van machinaal vertalen en meertalige information retrieval. Op dit moment wordt er veel onderzoek gedaan naar automatische selectie en generatie van correcte transliteraties, onder andere in het kader van jaarlijkse competities (Li et al. 2009, Kumaran et al. 2010b, Zhang et al. 2011). In dit proefschrift bespreken we twee versies van Dynamische Bayesiaanse Netwerken (DBN) toegepast op automatische selectie en generatie van transliteraties. De eerste versie gebruikt Pair Hidden Markovmodellen (Pair HMMs), een algemene versie van de klassieke Hidden Markovmodellen. De tweede versie is geïnspireerd op spraakherkenning en gebruikt DBNsjablonen voor de definitie van verschillende klassen van DBNmodellen gebaseerd op transductie. Zowel de Pair HMMs als de transductieve DBNmodellen zijn met succes toegepast op taalverwerkingstaken die vergelijkbaar zijn met automatische transliteratie. Pair HMMs werkten goed voor de herkenning van etymologisch verwante woorden (Mackay and Kondrak 2005) en vergelijking van dialecten (Wieling et al. 2007). Transductieve DBNmodellen deden het goed bij het classificeren van uitspraak (Filali and Bilmes 2005) en eveneens bij herkenning van etymologisch verwante woorden (Kondrak and Sherif 2006).

We onderzochten welke invloed het varieëren van de eigenschappen van DBNmodellen heeft op hun toepassing voor automatische transliteratie. We begonnen met de instellingen voor Pair HMMs die in de literatuur werden genoemd (Mackay

and Kondrak 2005, Wieling et al. 2007). Bij dit experiment gebruikten we onze eigen transliteratiedata bestaande uit paren van geografische namen verkregen uit de online Geonamesdatabase[1]. We kozen voor vier taalparen: Engels-Nederlands, Engels-Frans, Engels-Duits en English-Russisch. We wilden ook automatische transliteratie onderzoeken voor taalparen met hetzelfde schrift, iets dat voor zover wij weten niet eerder was gedaan. De eerste resultaten gaven aan dat het belangrijk was om Pair HMMs te gebruiken die de verschillen tussen brontaal en doeltaal goed konden modelleren. Zo bleek dat de emissieparameters van de Pair HMMs moesten worden gebaseerd op het schrift van de brontaal of dat van de doeltaal en niet op een gecombineerde versie hiervan.

Vervolgens hebben de invloed op de taak onderzocht van de transitieparameters van de Pair HMMs. Deze optionele parameters bepalen de kansen op een transitie van een edittoestand in de modellen naar een andere. Bij dit experiment gebruikten we data van de transliteratiecompetities van de edities van de Named Entities Workshop van 2009 en 2010, te weten de taalparen Engels-Bengaals, Engels-Chinees, Engels-Hindi, Engels-Kannada, Engels-Russisch, Engels-Tamil en Engels-Thai. De uitkomsten van dit experiment hebben we vergeleken met een standaardbaseline gebaseerd op n-grammen. Modellen die de transliteratieparameters gebruikten, bleken beter te werken dan modellen die ze niet gebruikten.

Daarna hebben we ook verschillende met Pair HMMs geassocieerde algoritmes vergeleken: zowel het forwardalgoritme en het Viterbialgoritme als hun log-oddsvariant in combinatie met een willekeurig geïnitialiseerde Pair HMM. De log-oddsversies leverden hogere transliteratiekwaliteit op dan de standaardversies van het forwardalgoritme en het Viterbialgoritme. Daarnaast weren de scores van de log-oddsversies niet beïnvloed door de waarden van de transitieparameters. De Pair HMMs haalden in de generatietaak voor zes van de zeven taalparen betere accuraatheden en Mean Reciprocal Rankscores dan de baseline. Deze resultaten geven aan dat Pair HMMS waardevol zijn bij het automatisch zoeken naar transliteraties.

Onze tweede methode voor automatische transliteratie gebruikt Dynamische Bayesiaanse Netwerken (DBN). Deze zijn gebaseerd op het werk van Ristad en Yianilos (1997), die een geheugenvrije stochastische transducer representeerden als een geheugenvrij context-onafhankelijk DBNmodel. We gebruiken de versie van dit model uit Filali en Bilmes (2005) voor het berekenen van de overeenkomst tussen woordparen en zetten het in voor het selecteren van transliteraties. Op basis van dit model maken we drie generalisaties van transductiegebaseerde DBNmodellen. Elk van deze generalisaties gebruikt een specifiek type van temporele afhankelijkheid waarvan we aannemen dat die belangrijk is voor automatische transliteratie. De drie afhankelijkheden zijn geheugenafhankelijkheid van de editoperaties, lengteafhankeli-

[1] http://www.geonames.org

jkheid van de editoperaties en tekencontextafhankelijkheden in de brontaalwoorden en de doeltaalwoorden. We hebben acht transductiegebaseerde DBNmodellen geëvalueerd op competitiedata uit 2009 en 2010 beschreven in de vorige alinea's. Ook de opzet van de experimenten was gelijk aan de selectie-experimenten met Pair HMMs.

Van de geëvalueerde DBNmodellen kwamen de contextgevoelige modellen als beste uit de bus. Deze modellen haalden ook voor vijf van de zeven taalparen betere scores dan de beste Pair HMMs. Daarnaast deden zij het beter dan de baseline voor alle zeven taalparen. Deze resultaten tonen het belang aan van het gebruik van context in DBNmodellen. Uit een analyse van de door de modellen gemaakte fouten bleek dat een beter prestatie mogelijk was als de keuzes van de verschillende modellen zouden worden gecombineerd. In een vervolgexperiment met combinatiemethodes gebaseerd op gemiddelde transliteratieafstand van de DBNmodellen bleken combinaties van modellen inderdaad betere accuraatheden en MRR-scores te bereiken dan individuele DBNmodellen.

Vervolgens hebben we enkele van onze Pair HMMs toegepast op transliteratiegeneratie en de resultaten vergeleken met die van de state-of-the-artmodellen uit de competities van 2009 en 2010 (Li et al. 2009, Li et al. 2010). Daarbij hebben we de parameters van de modellen gemodelleerd als gewogen transductieparameters die kandidaatwoorden genereerden op basis van woorden in de brontaal. In deze experimenten hebben we Weighted Finite State Transducers (WFST) ingezet als benaderingen van Pair HMMs. De prestaties van deze modellen deden onder voor de generatieresultaten van de state-of-the-artmodellen, zoals bijvoorbeeld die van phrase-based statistical machine translation (PSMT). Een nadeel van Pair HMM-gebaseerde WFST-modellen ten opzichte van deze PSMT-modellen is het onbreken van contextinformatie voor de bron- en doeltaal. Uit een foutanalyse bleek dat de WFST-modellen consequent dezelfde fouten maakten. Het toepassen van enkele transformatieregels op de uitvoer van de modellen leidde tot een grote prestatieverbetering, zij het nog niet tot de beoogde state-of-the-artresultaten.

Tenslotte pasten we enkele van onze Pair HMMs en contextgevoelige DBNmodellen toe op de data van de transliteratiedetectiecompetitie van Kumaran et al 2010a. Deze data bestond uit paren van titels van Wikipedia-artikelen voor de taalparen Engels-Arabisch, Engels-Chinees, Engels-Hindi, Engels-Russisch en Engels-Tamil. Zowel de Pair HMMs als de DBNmodellen haalden goede scores voor de taalparen Engels-Arabisch, Engels-Chinees en Engels-Russisch. Voor de taalparen Engels-Hindi en Engels-Tamil presteerden de Pair HMMs gebaseerd op log-odds zelfs beter dan de state-of-the-artmodellen gepresenteerd in Kumaran et al. (2010b).

Groningen Dissertations in Linguistics (GRODIL)

1. Henriëtte de Swart (1991). *Adverbs of Quantification: A Generalized Quantifier Approach.*
2. Eric Hoekstra (1991). *Licensing Conditions on Phrase Structure.*
3. Dicky Gilbers (1992). *Phonological Networks. A Theory of Segment Representation.*
4. Helen de Hoop (1992). *Case Configuration and Noun Phrase Interpretation.*
5. Gosse Bouma (1993). *Nonmonotonicity and Categorial Unification Grammar.*
6. Peter Blok (1993). *The Interpretation of Focus: an epistemic approach to pragmatics.*
7. Roelien Bastiaanse (1993). *Studies in Aphasia.*
8. Bert Bos (1993). *Rapid User Interface Development with the Script Language Gist.*
9. Wim Kosmeijer (1993). *Barriers and Licensing.*
10. Jan-Wouter Zwart (1993). *Dutch Syntax: A Minimalist Approach.*
11. Mark Kas (1993). *Essays on Boolean Functions and Negative Polarity.*
12. Ton van der Wouden (1994). *Negative Contexts.*
13. Joop Houtman (1994). *Coordination and Constituency: A Study in Categorial Grammar.*
14. Petra Hendriks (1995). *Comparatives and Categorial Grammar.*
15. Maarten de Wind (1995). *Inversion in French.*
16. Jelly Julia de Jong (1996). *The Case of Bound Pronouns in Peripheral Romance.*
17. Sjoukje van der Wal (1996). *Negative Polarity Items and Negation: Tandem Acquisition.*
18. Anastasia Giannakidou (1997). *The Landscape of Polarity Items.*
19. Karen Lattewitz (1997). *Adjacency in Dutch and German.*

20. Edith Kaan (1997). *Processing Subject-Object Ambiguities in Dutch.*
21. Henny Klein (1997). *Adverbs of Degree in Dutch.*
22. Leonie Bosveld-de Smet (1998). *On Mass and Plural Quantification: The Case of French 'des'/'du'-NPs.*
23. Rita Landeweerd (1998). *Discourse Semantics of Perspective and Temporal Structure.*
24. Mettina Veenstra (1998). *Formalizing the Minimalist Program.*
25. Roel Jonkers (1998). *Comprehension and Production of Verbs in Aphasic Speakers.*
26. Erik F. Tjong Kim Sang (1998). *Machine Learning of Phonotactics.*
27. Paulien Rijkhoek (1998). *On Degree Phrases and Result Clauses.*
28. Jan de Jong (1999). *Specific Language Impairment in Dutch: Inflectional Morphology and Argument Structure.*
29. H. Wee (1999). *Definite Focus.*
30. Eun-Hee Lee (2000). *Dynamic and Stative Information in Temporal Reasoning: Korean Tense and Aspect in Discourse.*
31. Ivilin Stoianov (2001). *Connectionist Lexical Processing.*
32. Klarien van der Linde (2001). *Sonority Substitutions.*
33. Monique Lamers (2001). *Sentence Processing: Using Syntactic, Semantic, and Thematic Information.*
34. Shalom Zuckerman (2001). *The Acquisition of "Optional" Movement.*
35. Rob Koeling (2001). *Dialogue-Based Disambiguation: Using Dialogue Status to Improve Speech Understanding.*
36. Esther Ruigendijk (2002). *Case Assignment in Agrammatism: a Cross-linguistic Study.*
37. Tony Mullen (2002). *An Investigation into Compositional Features and Feature Merging for Maximum Entropy-Based Parse Selection.*
38. Nanette Bienfait (2002). *Grammatica-onderwijs aan allochtone jongeren.*
39. Dirk-Bart den Ouden (2002). *Phonology in Aphasia: Syllables and Segments in Level-specific Deficits.*
40. Rienk Withaar (2002). *The Role of the Phonological Loop in Sentence Comprehension.*
41. Kim Sauter (2002). *Transfer and Access to Universal Grammar in Adult Second Language Acquisition.*
42. Laura Sabourin (2003). *Grammatical Gender and Second Language Processing: An ERP Study.*
43. Hein van Schie (2003). *Visual Semantics.*
44. Lilia Schürcks-Grozeva (2003). *Binding and Bulgarian.*
45. Stasinos Konstantopoulos (2003). *Using ILP to Learn Local Linguistic Structures.*
46. Wilbert Heeringa (2004). *Measuring Dialect Pronunciation Differences using Levenshtein Distance.*
47. Wouter Jansen (2004). *Laryngeal Contrast and Phonetic Voicing: A Laboratory Phonology Approach to English, Hungarian and Dutch.*

48. Judith Rispens (2004). *Syntactic and Phonological Processing in Developmental Dyslexia.*
49. Danielle Bougaïré (2004). *L'approche communicative des campagnes de sensibilisation en santé publique au Burkina Faso: les cas de la planification familiale, du sida et de l'excision.*
50. Tanja Gaustad (2004). *Linguistic Knowledge and Word Sense Disambiguation.*
51. Susanne Schoof (2004). *An HPSG Account of Nonfinite Verbal Complements in Latin.*
52. M. Begoña Villada Moirón (2005). *Data-driven identification of fixed expressions and their modifiability.*
53. Robbert Prins (2005). *Finite-State Pre-Processing for Natural Language Analysis.*
54. Leonoor van der Beek (2005). *Topics in Corpus-Based Dutch Syntax.*
55. Keiko Yoshioka (2005). *Linguistic and gestural introduction and tracking of referents in L1 and L2 discourse.*
56. Sible Andringa (2005). *Form-focused instruction and the development of second language proficiency.*
57. Joanneke Prenger (2005). *Taal telt! Een onderzoek naar de rol van taalvaardigheid en tekstbegrip in het realistisch wiskundeonderwijs.*
58. Neslihan Kansu-Yetkiner (2006). *Blood, Shame and Fear: Self-Presentation Strategies of Turkish Women's Talk about their Health and Sexuality.*
59. Mónika Z. Zempléni (2006). *Functional imaging of the hemispheric contribution to language processing.*
60. Maartje Schreuder (2006). *Prosodic Processes in Language and Music.*
61. Hidetoshi Shiraishi (2006). *Topics in Nivkh Phonology.*
62. Tamás Biró (2006). *Finding the Right Words: Implementing Optimality Theory with Simulated Annealing.*
63. Dieuwke de Goede (2006). *Verbs in Spoken Sentence Processing: Unraveling the Activation Pattern of the Matrix Verb.*
64. Eleonora Rossi (2007). *Clitic production in Italian agrammatism.*
65. Holger Hopp (2007). *Ultimate Attainment at the Interfaces in Second Language Acquisition: Grammar and Processing.*
66. Gerlof Bouma (2008). *Starting a Sentence in Dutch: A corpus study of subject- and object-fronting.*
67. Julia Klitsch (2008). *Open your eyes and listen carefully. Auditory and audiovisual speech perception and the McGurk effect in Dutch speakers with and without aphasia.*
68. Janneke ter Beek (2008). *Restructuring and Infinitival Complements in Dutch.*
69. Jori Mur (2008). *Off-line Answer Extraction for Question Answering.*
70. Lonneke van der Plas (2008). *Automatic Lexico-Semantic Acquisition for Question Answering.*
71. Arjen Versloot (2008). *Mechanisms of Language Change: Vowel reduction in 15th century West Frisian.*
72. Ismail Fahmi (2009). *Automatic term and Relation Extraction for Medical Question Answering System.*

73. Tuba Yarbay Duman (2009). *Turkish Agrammatic Aphasia: Word Order, Time Reference and Case.*
74. Maria Trofimova (2009). *Case Assignment by Prepositions in Russian Aphasia.*
75. Rasmus Steinkrauss (2009). *Frequency and Function in WH Question Acquisition. A Usage-Based Case Study of German L1 Acquisition.*
76. Marjolein Deunk (2009). *Discourse Practices in Preschool. Young Children's Participation in Everyday Classroom Activities.*
77. Sake Jager (2009). *Towards ICT-Integrated Language Learning: Developing an Implementation Framework in terms of Pedagogy, Technology and Environment.*
78. Francisco Dellatorre Borges (2010). *Parse Selection with Support Vector Machines.*
79. Geoffrey Andogah (2010). *Geographically Constrained Information Retrieval.*
80. Jacqueline van Kruiningen (2010). *Onderwijsontwerp als conversatie. Probleemoplossing in interprofessioneel overleg.*
81. Robert G. Shackleton (2010). *Quantitative Assessment of English-American Speech Relationships.*
82. Tim Van de Cruys (2010). *Mining for Meaning: The Extraction of Lexico-semantic Knowledge from Text.*
83. Therese Leinonen (2010). *An Acoustic Analysis of Vowel Pronunciation in Swedish Dialects.*
84. Erik-Jan Smits (2010). *Acquiring Quantification. How Children Use Semantics and Pragmatics to Constrain Meaning.*
85. Tal Caspi (2010). *A Dynamic Perspective on Second Language Development.*
86. Teodora Mehotcheva (2010). *After the fiesta is over. Foreign language attrition of Spanish in Dutch and German Erasmus Students.*
87. Xiaoyan Xu (2010). *English language attrition and retention in Chinese and Dutch university students.*
88. Jelena Prokić (2010). *Families and Resemblances.*
89. Radek Šimík (2011). *Modal existential wh-constructions.*
90. Katrien Colman (2011). *Behavioral and neuroimaging studies on language processing in Dutch speakers with Parkinson's disease.*
91. Siti Mina Tamah (2011). *A Study on Student Interaction in the Implementation of the Jigsaw Technique in Language Teaching.*
92. Aletta Kwant (2011). *Geraakt door prentenboeken. Effecten van het gebruik van prentenboeken op de sociaal-emotionele ontwikkeling van kleuters.*
93. Marlies Kluck (2011). *Sentence amalgamation.*
94. Anja Schüppert (2011). *Origin of asymmetry: Mutual intelligibility of spoken Danish and Swedish.*
95. Peter Nabende (2011). *Applying Dynamic Bayesian Networks in Transliteration Detection and Generation.*

Grodil
Secretary of the Department of General Linguistics
Postbus 716
9700 AS Groningen
The Netherlands